



SOBRE GRAFOS DE CAYLEY, PERMUTAÇÕES E CIRCUITOS REVERSÍVEIS

André da Cunha Ribeiro

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Celina Miraglia Herrera de
Figueiredo
Luis Antonio Brasil Kowada

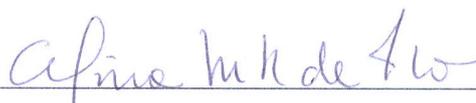
Rio de Janeiro
Maio de 2013

SOBRE GRAFOS DE CAYLEY, PERMUTAÇÕES E CIRCUITOS
REVERSÍVEIS

André da Cunha Ribeiro

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE)
DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR
EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:



Prof^a. Celina Miraglia Herrera de Figueiredo, D.Sc.



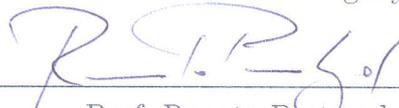
Prof. Luis Antonio Brasil Kowada, D.Sc.



Prof. Franklin de Lima Marquezino, D.Sc.



Prof^a. Diane Castonguay, Ph.D.



Prof. Renato Portugal, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 2013

Ribeiro, André da Cunha

Sobre Grafos de Cayley, Permutações e Circuitos Reversíveis/André da Cunha Ribeiro. – Rio de Janeiro: UFRJ/COPPE, 2013.

XII, 80 p.: il.; 29, 7cm.

Orientadores: Celina Miraglia Herrera de Figueiredo

Luis Antonio Brasil Kowada

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 77 – 80.

1. Grafo de Cayley. 2. Ciclo hamiltoniano. 3. Conjectura Lovász. 4. Diâmetro. 5. Transposições pré-fixadas unitárias. 6. Rearranjo de genoma. 7. Circuitos reversíveis. 8. Computações reversíveis. 9. Portas quânticas. I. Figueiredo, Celina Miraglia Herrera de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*dedico este trabalho aos meus
Pais, Rita e Cilênio, a
Conceição Carolina da Cunha e
Sebastião Ribeiro de Moraes.*

Agradecimentos

Agradeço a Deus, em primeiro lugar, pelo dom da vida. À minha esposa Vanessa, minha companheira e amiga, que mesmo longe sempre esteve ao meu lado me apoiando, acalmando e encorajando. Que soube entender as minhas ausências e, em certos momentos, a minha impaciência. Aos meus familiares, principalmente aos meus pais, Cilênio e Rita, minha irmã Andréia e meus sobrinhos, Guilherme e Cilênio Neto, que sempre incentivaram e torceram por mim. Aos professores Fabiano e Rommel pelas ações que iniciaram este trabalho. De maneira especial, agradeço aos professores Celina e Kowada pelo ensino, pelo apoio e pela paciência que sempre demonstraram ter. Não tenho palavras para expressar minha gratidão. Aos membros da banca examinadora, os professores Diane, Franklin, Daniel e Renato por aceitarem o desafio de participar deste trabalho. Aos amigos que fiz no Rio de Janeiro, que foram/são uma extensão da minha família. Em especial aos meus “irmãos” Luisinho, Diana e Hélio e aos meus “primos” Cabessa, Hugo Cunha e Daniel Posner. A todos os meus amigos conterrâneos, que sempre torceram e acreditaram em mim. Em especial, aos “cumpanheiros” Carmem, Erika e Herbert por sempre me acolherem. Aos amigos, professores e funcionários do PESC/COPPE e da linha de Algoritmos e Combinatória. Aos amigos, professores e funcionários do IF Goiano. À CAPES pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

SOBRE GRAFOS DE CAYLEY, PERMUTAÇÕES E CIRCUITOS REVERSÍVEIS

André da Cunha Ribeiro

Maio/2013

Orientadores: Celina Miraglia Herrera de Figueiredo

Luis Antonio Brasil Kowada

Programa: Engenharia de Sistemas e Computação

Neste trabalho, apresentamos propriedades importantes de alguns grafos de Cayley associados aos problemas: ciclo hamiltoniano, interconexão de redes, rearranjo de genomas e circuito reversível.

Em particular, mostramos que os grafos $H_{l,p}$ e $H'_{l,p}$ são grafos de Cayley e têm ciclo hamiltoniano, o que corrobora a conjectura de Lovász. Estabelecemos que o grafo $H'_{l,p}$ tem grau $(2l - 2)$ e diâmetro $(\lfloor \frac{p}{2} \rfloor (l - 1))$, e o grafo $H_{l,p}$ tem grau $(l^2 - l)$ e o diâmetro é calculado por um algoritmo com complexidade de tempo de $O(l)$. As propriedades estabelecidas suportam que os grafos $H_{l,p}$ e $H'_{l,p}$ são bons esquemas para interconexão de redes.

O grafo de rearranjo por transposições pré-fixadas unitárias é formado pelo conjunto de vértices que são as permutações do grupo simétrico S_n , e pelo conjunto de arestas obtido da seguinte forma: dois vértices são adjacentes se existe uma transposição pré-fixada unitária que, aplicada a uma permutação, gera a outra. Apresentamos propriedades deste grafo de Cayley, entre as principais mostramos que o diâmetro é exatamente $n - 1$ e a existência de ciclo hamiltoniano, que corrobora a conjectura de Lovász sobre os caminhos hamiltonianos nos grafos vértice-transitivos.

Usando grafos de Cayley desenvolvemos uma estrutura para ajudar na análise das contagem de portas e dos custos quânticos resultantes das sínteses de circuitos reversíveis. Apresentamos um algoritmo para a síntese de circuitos reversíveis com a complexidade de $\lfloor \frac{2}{3} 2^n (n - \frac{2}{3}) \rfloor + 1$ portas Toffoli de controles mistos. Além disso, temos que o limite inferior para o diâmetro do grafo de Cayley é $n2^{n-1}$.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ON CAYLEY GRAPHS, PERMUTATIONS AND REVERSIBLE CIRCUITS

André da Cunha Ribeiro

May/2013

Advisors: Celina Miraglia Herrera de Figueiredo

Luis Antonio Brasil Kowada

Department: Systems Engineering and Computer Science

In this work, we present some important properties of Cayley graphs associated with problems: Hamiltonian cycle, Interconnection Networks, Genome Rearrangement and Reversible Circuit.

In particular, we show that graphs $H_{l,p}$ and $H'_{l,p}$ are Cayley graphs and have a hamiltonian cycle, which corroborates to Lovász conjecture. We establish that the graph $H'_{l,p}$ has degree $(2l - 2)$ and diameter $(\lfloor \frac{p}{2} \rfloor (l - 1))$ and the graph $H_{l,p}$ has degree $(l^2 - l)$ and the diameter can be calculated by an algorithm of time $O(l)$. The established properties support the graphs $H_{l,p}$ and $H'_{l,p}$ to be good schemes of interconnection networks.

The Unitary Prefix Transposition Rearrangement Graph has the vertex set as the permutations in the Symmetric Group S_n and the edge set obtained as follows: two vertices are adjacent if there exists a unitary prefix transposition that applied to a permutation produces the other one. We present properties of this Cayley graph, among which the main ones are the exact value of the diameter to be $n - 1$ and the existence of hamiltonian cycles which corroborates to Lovász conjecture about hamiltonian cycles in Cayley graphs.

We also develop a theory of Cayley graphs which can be used as a framework to analyse gate counts and quantum costs resulting from this reversible circuit synthesis. We present an algorithm for reversible circuit synthesis with complexity of $\lfloor \frac{2}{3} 2^n (n - \frac{2}{3}) \rfloor + 1$ mixed-control Toffoli gates. In addition, the Cayley graph has a lower bound for the diameter of $n2^{n-1}$.

Sumário

Lista de Figuras	x
Lista de Tabelas	xii
1 Introdução	1
1.1 Conceitos Básicos da Teoria dos Grafos	4
1.2 Grafos de Cayley	7
1.2.1 Famílias de grafos de Cayley	9
2 Grafos $H_{l,p}$ e $H'_{l,p}$	11
2.1 Grafo $H_{l,p}$	11
2.2 Grafo $H'_{l,p}$	18
2.3 Ciclo hamiltoniano	20
2.3.1 Grafos de Cayley abeliano	20
2.3.2 Ciclo hamiltoniano com código Gray	22
2.4 Diâmetro	24
2.4.1 Grafo $H_{l,p}$	25
2.4.2 Grafo $H'_{l,p}$	32
2.5 Grafo $GC_n(b)$	33
3 Transposições Pré-fixadas Unitárias	36
3.1 Permutações	37
3.2 Grafo de Rearranjo por Transposições Pré-fixadas Unitárias	40
3.3 Diâmetro	43
3.4 Ciclos úteis no grafo $URG(n)$	44
3.5 Ciclo hamiltoniano no grafo $URG(n)$	46
4 Circuitos Reversíveis	48
4.1 Circuitos reversíveis e quânticos	48
4.2 Síntese de circuito usando portas G-Toffoli	54
4.3 Síntese de circuitos usando portas CM-Toffoli	57
4.3.1 Método Hipercubo	57

4.3.2	Método de controles mistos	62
4.4	Análise da síntese de circuito baseada no grafo de Cayley I_n	68
4.5	Análise da síntese de circuito baseada no grafo de Cayley M_n	70
5	Conclusões	74
5.1	Trabalhos futuros	76
	Referências Bibliográficas	77

Lista de Figuras

1.1	Grafo simples.	4
1.2	C_4	5
1.3	K_3	5
1.4	Grafos bipartidos.	5
1.5	Os grafos H e I são subgrafos induzidos de G	6
1.6	Os grafos H e I são subgrafos geradores de G	6
2.1	Grafo $H_{3,3}$ com repetição de vértices.	12
2.2	Grafo $H_{3,4}$ com repetição de vértices.	12
2.3	Grafo $H'_{3,3}$ com repetição de vértices.	19
2.4	$GC_3(2) \cong H'_{4,2}$	33
2.5	$GC_2(3) \cong H'_{3,3}$	34
3.1	Os grafos $URG(3)$ e $URG(4)$	37
3.2	Os grafos de Cayley $BS(4)$, $URG(4)$, $PRG(4)$ e $TRG(4)$	41
3.3	Parte do $URG(4)$ a partir do $URG(3)$ com todas as arestas incidentes ao vértice identidade ι	43
3.4	Um ciclo útil no grafo $URG(4)$. As arestas representantes associadas às $ut_r(3)$ são sólidas e as arestas ligantes associadas às $ut_l(4)$ são tracejadas.	44
3.5	Um ciclo útil no grafo $URG(5)$	45
4.1	Porta G-Toffoli representando $C^4NOT(a, b, c, d)$. A linha do topo denota o bit menos significativo.	49
4.2	Circuito representando a biblioteca de portas CNT	49
4.3	Porta CM-Toffoli representada por $C^4NOT(a', b, c', d)$. A linha do topo denota o bit menos significativo.	50
4.4	Decomposição de uma porta quântica com dois controles em uma sequência de portas quânticas com um único controle.	51
4.5	Decomposição de uma porta quântica com dois controles, sendo um controle negativo, em uma sequência de portas quânticas com um único controle.	51

4.6	Implementação da porta CM-Toffoli para $n = 6$ e três linhas de lixo baseada nos trabalhos de Kowada [24] e Maslov e Saeedi [30].	52
4.7	Implementação da porta CM-Toffoli para $n = 6$ e três linhas de lixo baseada nos trabalhos de Barenco <i>et al.</i> [4] e Maslov <i>et al.</i> [31].	53
4.8	Implementação da porta CM-Toffoli para $n = 8$ com uma linha de lixo baseada nos trabalhos de Barenco <i>et al.</i> [4] e Maslov <i>et al.</i> [31].	53
4.9	Decomposição da porta CM-Toffoli dada na Figura 4.8 em portas elementares de 3-bit.	54
4.10	Implementação da porta G-Toffoli para $n = 4$ e sem linhas de lixo baseada nos trabalhos de Barenco <i>et al.</i> [4] e Maslov <i>et al.</i> [31].	54
4.11	Circuito reversível que transforma permutação ι em $\pi = [1\ 0\ 3\ 2\ 5\ 7\ 4\ 6]$, de acordo com o exemplo da Tabela 4.2.	55
4.12	Circuito reversível que transforma permutação $\pi = [7\ 4\ 1\ 0\ 3\ 2\ 6\ 5]$, de acordo com o exemplo da Tabela 4.4.	58
4.13	Circuito reversível que transforma a permutação ι na permutação $\pi = [1\ 7\ 5\ 4\ 3\ 2\ 0\ 6]$, de acordo com o exemplo da Tabela 4.6.	64
4.14	Grafo I_2	69
4.15	Grafo M_2	73

Lista de Tabelas

2.1	Diâmetros do grafo $H_{l,p}$, para valores de l e p entre 2 e 10.	28
2.2	Vértice z do grafo $H_{5,7}$ com soma $s_{l_1} = 8$ para valores $l_1 = 2$ e $l_1 = 3$	29
4.1	Custo quântico de portas reversíveis.	52
4.2	Exemplo da aplicação do algoritmo MMD.	56
4.3	Número de funções reversíveis 3×3 para o Algoritmo 4.1 usando portas G -Toffoli e comparando com os melhores resultados a partir de Shende <i>et al.</i> [42].	56
4.4	Evolução da permutação π sendo transformada em ι pelo método Hiper cubo.	58
4.5	Número de funções reversíveis 3×3 para o Algoritmo 4.2 usando portas CM -Toffoli e comparando com os resultados ótimos.	61
4.6	Evolução da permutação π sendo transformada em ι pelo método de controles mistos.	63
4.7	Número de funções reversíveis 3×3 para o Algoritmo 4.4 usando portas CM -Toffoli e comparando com os resultados ótimos.	67
4.8	Análise do custo quântico para a síntese de circuito baseada no grafo de Cayley I_n	70
4.9	Análise do custo quântico para a síntese de circuito baseada no grafo de Cayley M_n	71
5.1	Resultados conhecidos dos grafos $TRG(n)$, $PRG(n)$, $URG(n)$ e $BS(n)$	75

Capítulo 1

Introdução

Os grafos de Cayley foram introduzidos por Arthur Cayley em 1878, com o intuito de associar a teoria dos grupos à teoria dos grafos. A teoria dos grupos estuda as estruturas algébricas conhecidas como grupos. Um grupo é um conjunto de elementos associado a uma operação que combina dois elementos para formar um terceiro que também pertence ao conjunto. Além disso, um grupo possui as seguintes propriedades: a operação do grupo é associativa, existe um elemento identidade e o inverso de cada elemento também está no grupo. A teoria dos grafos estuda as estruturas discretas que modelam as relações entre os objetos de um determinado conjunto. Assim, um grafo é definido por um conjunto de vértices ou nós e um conjunto de arestas que conectam pares de vértices. Um grafo de Cayley é um grafo formado por um grupo e um conjunto gerador do grupo, onde os vértices estão associados aos elementos do grupo e as arestas estão associadas aos elementos do conjunto gerador. Os grafos de Cayley são conexos, regulares e vértice-transitivos.

Em 1970, Lovász conjecturou que todo grafo finito, conexo e vértice-transitivo possui caminho hamiltoniano [28]. Até o presente momento, apenas quatro grafos vértice-transitivos, com mais de dois vértices, mas sem ciclos hamiltonianos são conhecidos [23], porém estes quatro grafos têm caminhos hamiltonianos. No entanto, uma vez que nenhum destes quatro grafos são grafos de Cayley, podemos definir a conjectura de Lovász como todo grafo de Cayley, com mais de dois vértices tem um ciclo hamiltoniano. Neste sentido, apresentamos algumas famílias onde encontramos ciclos hamiltonianos, corroborando a conjectura de Lovász.

Uma questão que continua aberta para o problema de ciclo hamiltoniano nos grafos de Cayley do grupo simétrico foi introduzida por Jiang e Ruskey [19]. Eles questionaram se é possível encontrar um algoritmo que encontre um ciclo hamiltoniano nos grafos de Cayley do grupo simétrico S_n com complexidade $O(n!)$ mas utilizando memória da ordem de $O(n)$.

O grafo de Cayley do grupo simétrico S_n de todas as permutações de tamanho n é utilizado nos estudos de rearranjo de genomas. Nesta abordagem estamos interes-

sados em determinar a distância entre dois genomas diferentes através da aplicação de mutações no primeiro genoma, de modo que ao final da mutação, tenha dado origem ao segundo genoma. Um evento mutacional conhecido em rearranjo de genomas é o evento de transposição. Uma transposição é a operação que troca dois blocos contíguos de posições em uma permutação. O problema de ordenação por transposições consiste em determinar o menor número de eventos de transposição que transforme uma permutação na identidade, permutação com todos os elementos em ordem crescente. O problema de ordenação por transposições foi provado ser \mathcal{NP} -difícil por Bulteau *et al.* em 2010 e com publicação em 2012 [7].

O campo da computação reversível é motivado por várias fontes, tais como: processamento de sinais, criptografia, computação gráfica e circuitos fotônicos [41]. Um dos aspectos mais importantes do modelo de circuitos quânticos é a reversibilidade, que é uma consequência do postulado da evolução da mecânica quântica. De acordo com este postulado, o tempo de evolução no estado de um sistema quântico fechado é descrito por um operador unitário [34]. Em 1980, Toffoli mostrou que a teoria da computação reversível é um dos fundamentos da computação quântica. É também interessante notar que, em qualquer circuito reversível—clássico ou quântico—a saída contém informações suficientes para reconstruir a entrada, isto é, nenhuma informação da entrada é apagada [43].

Um conjunto de portas reversíveis é necessário para projetar os circuitos reversíveis. Devos *et al.* e Maslov *et al.* utilizaram a teoria dos grupos como ferramenta de análise das portas lógicas e dos conjuntos gerador do grupo de portas reversíveis [11]. Nesta tese, utilizamos os grafos de Cayley associados ao grupo simétrico S_{2^n} para analisarmos a síntese de circuitos reversíveis. Cada método de síntese proposto é composto por uma biblioteca diferente, cujas portas estão associadas ao conjunto gerador de um determinado grafo de Cayley.

Propriedades dos grafos de Cayley—como grau, distância, e diâmetro—são consideradas. O grau do grafo de Cayley é exatamente o tamanho do conjunto gerador, que por sua vez corresponde ao número de portas da biblioteca. A distância de dois vértices no grafo corresponde ao tamanho de um circuito ótimo—cada porta produz uma aresta no grafo de Cayley, de modo que o tamanho do circuito, com menor número de portas possível, corresponde à distância. Em relação ao diâmetro, uma propriedade importante dos grafos de Cayley é que o limite inferior para o seu diâmetro é também um limite inferior para o pior caso de qualquer algoritmo que utiliza a biblioteca de portas correspondente.

Esta tese contribui para os problemas de distância, diâmetro e ciclo hamiltoniano em algumas famílias de grafos de Cayley. Algoritmos polinomiais para estes problemas desafiadores são apresentados, o que consideramos como os resultados principais deste trabalho.

Esta tese está organizada da seguinte forma. Na Seção 1.1 apresentamos as definições usuais da teoria dos grafos que é destinada aos leitores que ainda não estão familiarizados com tal teoria. Em seguida, apresentamos na Seção 1.2 os grafo de Cayley, assim como estabelecemos algumas notações que são necessárias no decorrer deste trabalho.

No Capítulo 2, apresentamos os grafos $H_{l,p}$ e $H'_{l,p}$. Na Seção 2.1, definimos o grafo $H_{l,p}$ e mostramos que ele é um grafo de Cayley. Na Seção 2.2, apresentamos o grafo de Cayley $H'_{l,p}$, que é um grafo de Cayley do mesmo grupo do grafo $H_{l,p}$ mas com um conjunto gerador diferente, neste caso temos um conjunto gerador com menos elementos. Na Seção 2.3, mostramos duas formas diferentes de encontrar ciclos hamiltonianos nos grafos $H_{l,p}$ e $H'_{l,p}$. Na Seção 2.4, encontramos o diâmetro dos grafos $H_{l,p}$ e $H'_{l,p}$. Na Seção 2.5, mostramos que os grafos $H'_{3,p}$ são isomorfos aos grafos $GC_2(p)$.

No Capítulo 3, apresentamos a operação de transposição pré-fixada unitária. Na Seção 3.1 mostramos algumas definições sobre permutações e um exemplo de um grafo de Cayley do grupo simétrico S_n . Este grafo é utilizado nos estudos de ordenação por transposições pré-fixadas unitárias. Na Seção 3.2, provamos que o grafo de rearranjo por transposição pré-fixada unitária $URG(n)$ é um grafo de Cayley e consiste em n cópias isomórficas do grafo $URG(n - 1)$. Na Seção 3.3, mostramos que o diâmetro do grafo $URG(n)$ é $n - 1$. Na Seção 3.4, provamos a existência de n ciclos úteis que serão estendidos na Seção 3.5 para o ciclo hamiltoniano.

No Capítulo 4, mostramos um novo algoritmo de síntese de circuitos reversíveis e apresentamos um formalismo teórico para os processos de síntese usando os grafos de Cayley. Na Seção 4.1, definimos circuitos reversíveis e quânticos. Na Seção 4.2, apresentamos o método de síntese de circuito usando portas Toffoli generalizadas. Na Seção 4.3, mostramos um novo método de síntese de circuitos usando portas Toffoli de controles mistos. Na Seção 4.4, apresentamos uma análise da síntese de circuitos baseada no grafo de Cayley I_n . Na Seção 4.5, apresentamos uma análise da síntese de circuitos baseada no grafo de Cayley M_n .

Por fim, no Capítulo 5, concluimos esta tese. Na Seção 5.1, terminamos este trabalho com a proposta de pesquisas futuras. Voltamos o nosso interesse para o estudo de algoritmos eficientes para encontrar ciclos hamiltonianos nos grafos de Cayley do grupo simétrico, algoritmos para determinar a menor distância nos grafos de Cayley e processos mais eficientes para a síntese de circuitos.

Os resultados desta tese foram divulgados assim: apresentações no *Latin American Workshop on Cliques in Graphs* em 2010 e 2012; resumo estendido publicado na revista *Matemática Contemporânea* em 2010 [40]; resumo estendido submetido para *Matemática Contemporânea* em 2013 [38]; apresentação de artigo completo no *IV Workshop-School on Quantum Computation and Information* em 2012 [37]; artigo

completo a ser submetido em setembro de 2013 para o *ACM Journal on Emerging Technologies in Computing Systems Special Issue on Reversible Computation* [39]; e artigo completo submetido para o *Discrete Applied Mathematics* em 2013 [36].

1.1 Conceitos Básicos da Teoria dos Grafos

Um *grafo* $G = (V, E)$ é um par de conjuntos, onde V é um conjunto não vazio e finito cujos elementos são chamados de *vértices* e E é um conjunto de pares de vértices não ordenados que são chamados de *arestas*. Dada uma aresta $vw \in E$ ou $(v, w) \in E$, os vértices v e w são denominados *extremidades* da aresta e dizemos que a aresta vw é *incidente* nos vértices v e w . Denotamos $V(G)$ como o conjunto de vértices de um grafo G e $E(G)$ como o conjunto de arestas de um grafo G .

Arestas múltiplas são arestas que têm as mesmas extremidades. Um *laço* é uma aresta da forma xx , ou seja, cujas extremidades são iguais. Um grafo é chamado de *simples* quando ele não possui arestas múltiplas e nem laços. Neste trabalho, são considerados apenas grafos simples.

Seja $G = (V, E)$ um grafo qualquer. A cardinalidade do conjunto de vértices de G é chamada de *ordem do grafo* e é denotada por $|V|$. A cardinalidade do conjunto de arestas de G é denotada por $|E|$.

Podemos visualizar um grafo através de uma *representação geométrica* associando um ponto para cada vértice sobre uma superfície e para cada aresta vw uma curva ligando v a w . Se dois vértices são conectados por uma aresta, então eles são chamados de *adjacentes*. A Figura 1.1 ilustra uma representação geométrica de um grafo. Um grafo é *planar* se existe uma representação geométrica do grafo no plano sem cruzamento de arestas.

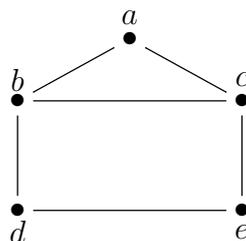


Figura 1.1: Grafo simples.

Os *vizinhos* ou a *vizinhança* de um vértice x é o conjunto de todos os vértices adjacentes a x . O conjunto de vizinhos de um vértice x é denotado por $N_G(x)$ ou simplesmente $N(x)$. Por exemplo, no grafo da Figura 1.1 temos $N(b) = \{a, c, d\}$.

O *grau* de um vértice x é o número de arestas que incidem em x , denotado por $d_G(x)$ ou simplesmente $d(x)$. O *grau mínimo* de um grafo é denotado por $\delta(G)$ e o *grau máximo* por $\Delta(G)$. Na Figura 1.1, temos $d(b) = 3$, $\delta(G) = 2$ e $\Delta(G) = 3$.

Algumas classes de grafos recebem nomes especiais. Uma classe é formada pelos grafos regulares. Um grafo $G = (V, E)$ é *regular* se todos os vértices de $V(G)$ têm o mesmo grau, ou seja, quando $\delta(G) = \Delta(G)$. Um grafo é chamado *k-regular* se $\delta(G) = \Delta(G) = k$. Na Figura 1.2, temos um grafo 2-regular. Um grafo conexo e 2-regular com n vértices é chamado de ciclo e é denotado por C_n .

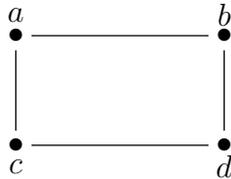


Figura 1.2: C_4 .

Outra classe especial de grafos é formada pelos grafos completos. Um grafo $G = (V, E)$ é *completo*, se todos os pares de vértices distintos são adjacentes. Um grafo completo com n vértices é denotado por K_n . Os grafos completos são muito importantes, pois vários resultados sobre grafos podem ser aplicados a eles. Na Figura 1.3, temos como exemplo o grafo completo com 3 vértices.

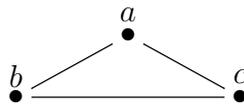
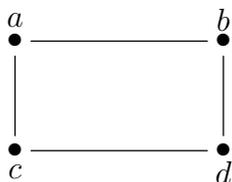
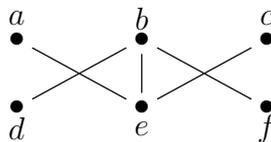


Figura 1.3: K_3 .

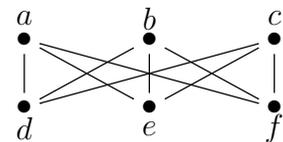
Uma classe especial de grafos é formada pelos que não possuem ciclos ímpares, chamados de grafos bipartidos. Um grafo $G = (V, E)$ é *bipartido* se existe uma partição do conjunto de todos os vértices em dois subconjuntos X e Y , tal que nenhuma aresta do grafo tenha ambas as extremidades em X ou ambas as extremidades em Y . Outra característica importante dos grafos bipartidos é o fato de que estes grafos podem ser coloridos com apenas duas cores, ou seja, uma cor para cada partição de vértice.



(a) Grafo $K_{2,2}$



(b) Grafo bipartido



(c) Completo $K_{3,3}$

Figura 1.4: Grafos bipartidos.

Um grafo é *bipartido completo*, com partição $\{X, Y\}$, se todos os vértices da parte X estão ligados a todos os vértices da parte Y . Ele é denotado por $K_{s,t}$, onde $s = |X|$ e $t = |Y|$. A Figura 1.4 ilustra grafos bipartidos.

Um grafo H é um *subgrafo* do grafo G , tal que o conjunto de vértices de H esteja contido no conjunto de vértices de G e o conjunto de arestas de H esteja contido no conjunto de arestas de G . Um subgrafo é obtido através da remoção de vértices ou de arestas. Temos tipos especiais de subgrafos quando removemos somente vértices ou somente arestas. No primeiro caso temos um subgrafo induzido e no segundo caso temos um subgrafo gerador, como segue.

Um grafo H é um *subgrafo induzido* do grafo G se, e somente se, $V(H) \subseteq V(G)$ e $E(H) = \{(x, y) \in E(G) \mid x \in V(H) \text{ e } y \in V(H)\}$, ou seja, as únicas remoções permitidas de G para obter H são as remoções de vértices e suas arestas incidentes. Se v é um vértice de G , então $G - v$ é o subgrafo induzido de G , obtido pela remoção do vértice v . Os grafos das Figuras 1.5(b) e 1.5(c) são subgrafos induzidos do grafo da Figura 1.5(a).

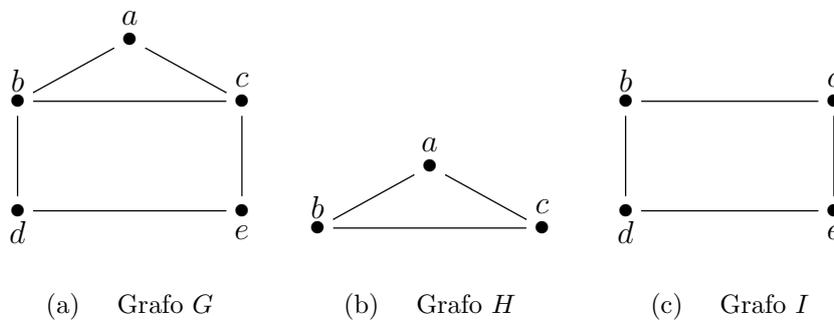


Figura 1.5: Os grafos H e I são subgrafos induzidos de G .

Um grafo H é um *subgrafo gerador* do grafo G se, e somente se, H for um subgrafo de G com todos os vértices de G , ou seja, H e G têm os mesmos vértices e as únicas remoções permitidas de G para obter H são as remoções de arestas. Se e é uma aresta de G , então $G - e$ é o subgrafo gerador de G , obtido pela remoção da aresta e . Os grafos H e I , das Figuras 1.6(b) e 1.6(c) são subgrafos geradores do grafo da Figura 1.6(a).

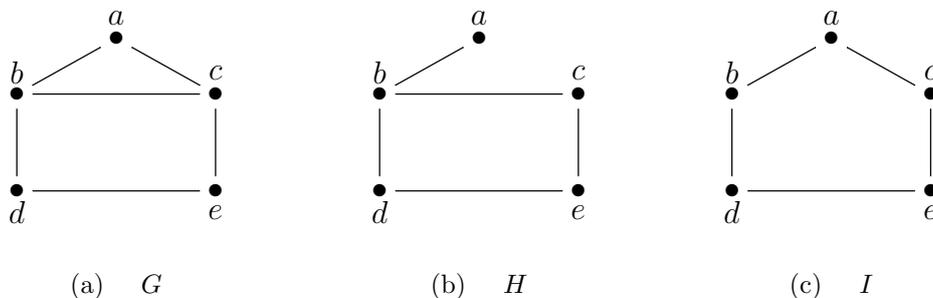


Figura 1.6: Os grafos H e I são subgrafos geradores de G .

Pela sua importância em teoria dos grafos, destacamos as definições referentes a

sequências de vértices e arestas. Um *caminho* entre dois vértices x e y de um grafo G é uma sequência de vértices $P = (v_1, v_2, v_3, \dots, v_k)$ onde, $x = v_1$, $y = v_k$, e $v_i v_{i+1}$ pertence às arestas do grafo, para todo $i = 1, \dots, k - 1$. Um caminho é denominado *simples* se os vértices que o constituem são todos distintos. Um caminho simples de k vértices é denotado por P_k , e seu comprimento é $k - 1$, relativo ao número de arestas.

Existe um tipo especial de caminho de comprimento maior ou igual a 3, em que o primeiro e o último vértices coincidem, este caminho é denominado de *ciclo*. Um *ciclo simples* é um ciclo onde todos os vértices que o constituem são distintos, a exceção do primeiro e do último vértices que coincidem. Um ciclo simples, sem arestas ligando dois vértices não consecutivos no ciclo, de k vértices é denotado por C_k , onde seu comprimento é k , dado pelo seu número de arestas.

Um *caminho hamiltoniano* é um caminho simples que contém todos os vértices do grafo. Um *ciclo hamiltoniano* é um caminho hamiltoniano onde existe uma aresta entre o primeiro e o último vértice.

Um grafo é *conexo*, se para todo par de vértices $\{v, w\}$, existe um caminho simples com extremos v e w . Caso contrário, o grafo é *desconexo*. Um subgrafo conexo H de um grafo G é *maximal*, se $H \cup \{u\}$ deixa de ser conexo para qualquer $u \in G - H$. Um *componente* de um grafo é um subgrafo conexo maximal. A partir da definição de grafo conexo e componente, podemos afirmar que um grafo é conexo se, e somente se, tiver um único componente. Se um grafo não possui arestas, então cada um de seus vértices constitui um componente conexo.

Sejam G um grafo conexo e $B \subseteq V(G)$ um subconjunto dos vértices de G . O conjunto B é chamado *separador* do grafo G , se $G - B$ tiver mais de um componente. A *conectividade* de G , $\kappa(G)$, é o tamanho do menor conjunto separador B . Se G não possui um conjunto separador, então $\kappa(G) = n - 1$, este é o caso dos grafos completos.

Um *automorfismo* de um grafo $G = (V, E)$ é um isomorfismo de G em G , isto é, uma função $f : G \rightarrow G$ que preserva a adjacência: $(u, v) \in E(G)$ se, e somente se $(f(u), f(v)) \in E(G)$. Dizemos que um grafo $G = (V, E)$ é *vértice-transitivo* se para todo par de vértices $u, v \in V(G)$ existe um automorfismo f , tal que $f(u) = v$. Todo grafo vértice-transitivo é regular.

1.2 Grafos de Cayley

Nesta seção, apresentamos alguns conceitos referentes a grupos e grafos de Cayley. Inicialmente, mostramos alguns conceitos fundamentais sobre grupos.

Definição 1.1 (Grupo). *Um grupo $(\mathcal{G}, +)$ é um conjunto não-vazio, munido de uma operação $+$, onde para todo $a, b, c \in \mathcal{G}$ temos as seguintes propriedades:*

1. existe o elemento identidade $\iota \in \mathcal{G}$, tal que $a + \iota = \iota + a = a$;
2. todo elemento possui um elemento inverso, isto é, para todo $a \in \mathcal{G}$, existe um $-a \in \mathcal{G}$, tal que $a + (-a) = (-a) + a = \iota$;
3. a operação $+$ é associativa, $(a + b) + c = a + (b + c)$;
4. a operação $+$ é fechada em \mathcal{G} , isto é, $a + b \in \mathcal{G}$.

Por exemplo, o conjunto \mathbb{Z} dos números inteiros com operação de adição é o grupo $(\mathbb{Z}, +)$. Neste grupo, o elemento identidade é o número 0 e o inverso do elemento a é o elemento $-a$.

A definição 1.1 pode ser vista por outras operações, por exemplo multiplicação.

Um grupo $(\mathcal{G}, +)$ é finito quando contiver um número finito de elementos. Podemos citar o grupo $(\mathbb{Z}_n, +)$, onde \mathbb{Z}_n é o conjunto dos números inteiros módulo n . Este grupo é finito pois $\mathbb{Z}_n = \{0, 1, \dots, n - 1\}$ contém uma quantidade finita de elementos. A seguir mostramos outras propriedades particulares de alguns grupos.

Definição 1.2 (Grupo abeliano). *Um grupo $(\mathcal{G}, +)$ é chamado de grupo abeliano (ou grupo comutativo) se $a + b = b + a$ para todo $a, b \in \mathcal{G}$.*

Como dito anteriormente, nem todos grupos são abelianos. Por exemplo, (A, \cdot) onde A é o conjunto das matrizes pertencentes a $\mathbb{R}^{n \times n}$. Agora definimos conjunto gerador de um grupo.

Definição 1.3 (Conjunto gerador). *Seja $(\mathcal{G}, +)$ um grupo finito com o elemento identidade, denotado por ι . Um subconjunto C do grupo é um conjunto gerador, se qualquer elemento de \mathcal{G} pode ser obtido a partir de elementos de C , por uma quantidade finita de aplicações da operação $+$.*

Sejam A e B dois conjuntos, e $A \setminus B$ o conjunto de todos os elementos de A que não estão em B . Um elemento $c \in C$ é dito *redundante*, se ele pode ser obtido como um produto de elementos em $C \setminus \{c\}$, caso contrário é *não redundante*. Se todo elemento no conjunto gerador C é não redundante, então C é chamado de *conjunto gerador minimal*.

Definição 1.4 (Grafo de Cayley). *Seja C um conjunto gerador para o grupo $(\mathcal{G}, +)$. Dizemos que um grafo direcionado $\Gamma(V, E)$ é um grafo de Cayley associado ao par (\mathcal{G}, C) , formado por um grupo \mathcal{G} e um conjunto gerador C , se existe uma bijeção mapeando cada vértice $v \in V$ em cada elemento do grupo $g \in \mathcal{G}$, de tal forma que os elementos do grupo g e $h \in \mathcal{G}$ estão ligados por uma aresta dirigida $(g, h) \in E$ se, e somente se existe $c \in C$, tal que $h = g + c$.*

Dizemos que Γ possui a *propriedade livre de identidade*, se $\iota \notin C$ e, portanto, não existem laços em Γ . Temos também que o grafo de Cayley Γ possui a *condição simétrica*, se $c \in C$ implica em $-c \in C$, então para toda aresta de g para $g+c$, existe também uma aresta de $g+c$ para $g = (g+c) - c$. O grafo de Cayley que possui as propriedades livre de identidade e condição simétrica é um grafo não-direcionado. Neste trabalho, consideramos somente grafos de Cayley não-direcionados.

Proposição 1.5. [23] *Seja C um conjunto gerador para o grupo $(\mathcal{G}, +)$. O grafo de Cayley $\Gamma(V, E)$ associado ao par (\mathcal{G}, C) tem as seguintes propriedades:*

- i. $\Gamma(V, E)$ é conexo, regular e de grau igual à cardinalidade de C ;
- ii. $\Gamma(V, E)$ é vértice-transitivo.

Demonstração. Por definição, temos que C é conjunto gerador do grupo $(\mathcal{G}, +)$, portanto $\Gamma(V, E)$ é conexo. Como C é simétrico, ou seja, $C = -C$ onde $-C = \{-c : c \in C\}$, então todo vértice no grafo $\Gamma(V, E)$ tem grau igual a $|C|$. Assim, o grafo $\Gamma(V, E)$ é regular de grau igual a cardinalidade de C . O grafo de Cayley $\Gamma(V, E)$ é vértice-transitivo, já que a permutação $\pi_g, g \in \mathcal{G}$, definida por $\pi_g(h) = gh$ para cada $h \in \mathcal{G}$ é um automorfismo. \square

Proposição 1.6. [5, 23] *Nem todo grafo vértice-transitivo é um grafo de Cayley.*

Demonstração. Um contra-exemplo é o grafo de Petersen. Ele é um grafo vértice-transitivo, mas não é um grafo de Cayley. O grafo de Petersen tem ordem 10, grau 3 e diâmetro 2, podemos verificar estas propriedades se examinarmos o par (\mathcal{G}, C) onde \mathcal{G} é um grupo com ordem 10 e C é um conjunto gerador com ordem 3. Existem apenas dois grupos não isomorfos de ordem 10, e verificando todos os conjuntos geradores C com 3 elementos com as propriedades livres de identidade e condição simétrica, descobre-se que cada um é um grafo de Cayley com diâmetro maior que 2. \square

1.2.1 Famílias de grafos de Cayley

Agora, mostramos algumas classes de grafos conhecidas que também são grafos de Cayley.

O *grafo completo* K_n é um grafo de Cayley no grupo aditivo \mathbb{Z}_n dos números inteiros modulo n com conjunto gerador formado por todos elementos diferentes de zero de \mathbb{Z}_n .

O *circulante* é um grafo de Cayley formado pelo par (\mathbb{Z}_n, C) onde $C \subseteq \mathbb{Z}_n$ é um conjunto gerador arbitrário. Um exemplo simples de um circulante é o grafo C_n , tomando $C = \{1, n-1\}$.

O *toro multidimensional* T_k^n , com $n \geq 2$ e $k \geq 2$, é o grafo obtido pelo produto cartesiano de n ciclos de tamanho k . Ele consiste em k^n vértices de grau $2n$ e tem diâmetro $n \lfloor k/2 \rfloor$. Ele é um grafo de Cayley do grupo \mathbb{Z}_k^n com conjunto gerador formado por $2n$ elementos do conjunto $C = \{(c_1, \dots, c_n) : \text{existe } i \in \{1, \dots, n\}, \text{ tal que } c_i = 1 \text{ ou } c_i = -1 \text{ e para todo } k \in \{1, \dots, n\} - \{i\}, \text{ temos que } c_k = 0\}$.

O *hipercubo* H_n é um grafo de Cayley com conjunto de vértices $(x_1, x_2, \dots, x_n, |x_i \in \{0, 1\})$ em que dois vértices (v_1, v_2, \dots, v_n) e (u_1, u_2, \dots, u_n) são adjacentes se, e somente se $v_i = u_i$ para todos os pares v_i, u_i exceto um i , $1 \leq i \leq n$. Ele é um grafo de Cayley de ordem 2^n , grau n e diâmetro n . O hipercubo é considerado um caso particular do toro, ou seja, $H_n = T_2^n$. É um grafo de Cayley do grupo \mathbb{Z}_2^n com conjunto gerador formado por $2n$ elementos do conjunto $C = \{(c_1, \dots, c_n) : \text{existe } i \in \{1, \dots, n\}, \text{ tal que } c_i = 1 \text{ ou } c_i = -1 \text{ e para todo } k \in \{1, \dots, n\} - \{i\}, \text{ temos que } c_k = 0\}$.

Capítulo 2

Grafos $H_{l,p}$ e $H'_{l,p}$

Os grafos de Cayley são conexos, regulares, podem ter diâmetro logarítmico no número de vértices e podem ser utilizados para criar interconexão de redes. Um grafo geralmente utilizado para interconexão de redes é o grafo hipercubo [23, 44]. Neste capítulo, mostramos que os grafos da família $H_{l,p}$, definidos no contexto de partições de arestas, são grafos de Cayley e admitem ciclo hamiltoniano. Consideramos duas famílias de grafos de Cayley: grafos $H_{l,p}$ e $H'_{l,p}$, onde $H'_{l,p}$ é um subgrafo gerador do $H_{l,p}$ mais esparsos e o grafo hipercubo é isomorfo ao grafo $H'_{n+1,2}$. Também, mostramos que os grafos $H'_{l,p}$ e $H_{l,p}$ têm diâmetro e grau logarítmicos.

2.1 Grafo $H_{l,p}$

O grafo $H_{l,p}$ foi definido como um grafo auxiliar no estudo do problema de particionamento das arestas [18]. O problema de partição de arestas em K_l 's, chamado de PE_l , considera a partição do conjunto de arestas de um grafo entrada em subconjuntos de arestas tais que cada subconjunto induz um grafo completo de l vértices.

O grafo $H_{l,p}$ é formado por vértices contendo l coordenadas com valores entre 0 e $p - 1$, tal que a soma dos l coordenadas seja equivalente a 0 mod p , com $p \in \mathbb{Z}_+$ e existe uma aresta entre dois vértices, quando há um par de coordenadas correspondentes onde os valores diferem por uma unidade.

Definição 2.1. [18] Para $l \geq 3$ e $p \geq 3$, o grafo $H_{l,p} = (V_{l,p}, E_{l,p})$ é:

- $V_{l,p} = \{x = (x_1, \dots, x_l) \in \mathbb{Z}_p^l : \sum_{i=1}^l x_i \equiv 0 \pmod{p}\};$
- $E_{l,p} = \{(x, y) : \text{existem } i, j \text{ que satisfazem } y_k \equiv_p x_k \text{ para todo } k \neq \{i, j\} \text{ e } (y_i \equiv_p x_i + 1, y_j \equiv_p x_j - 1 \text{ ou } y_i \equiv_p x_i - 1, y_j \equiv_p x_j + 1)\}.$

Nas Figuras 2.1 e 2.2, apresentamos duas representações com repetição de vértices para os grafos $H_{3,3}$ e $H_{3,4}$ respectivamente. Embora os grafos não sejam planares,

já que são grafos regulares de grau 6, o desenho apresentado é uma representação no toro sem cruzamentos de arestas. Podemos observar que para $l = 3$, temos uma “grade” de tamanho $p \times p$.

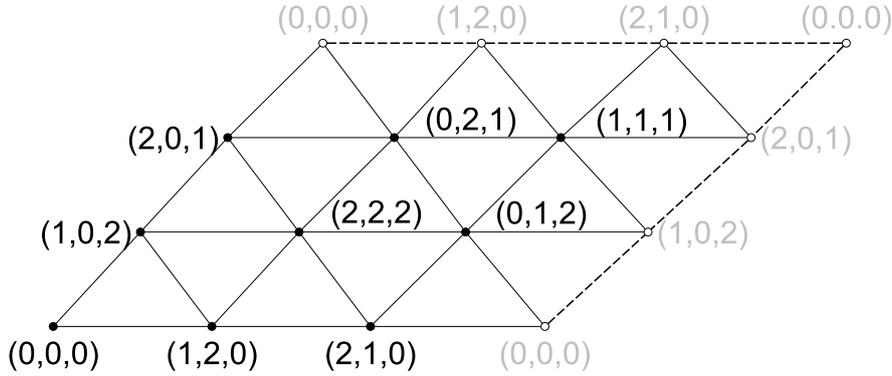


Figura 2.1: Grafo $H_{3,3}$ com repetição de vértices.

Lema 2.2. [18] O grafo $H_{l,p}$ tem as seguintes propriedades:

- a. o número total de vértices é $n = p^{l-1}$;
- b. o grau de cada vértice é $l(l-1)$;
- c. o número total de arestas é $\binom{l}{2}p^{l-1}$;
- d. se $\log_2 p > l$, então o grau é menor que $\log_2 n$.

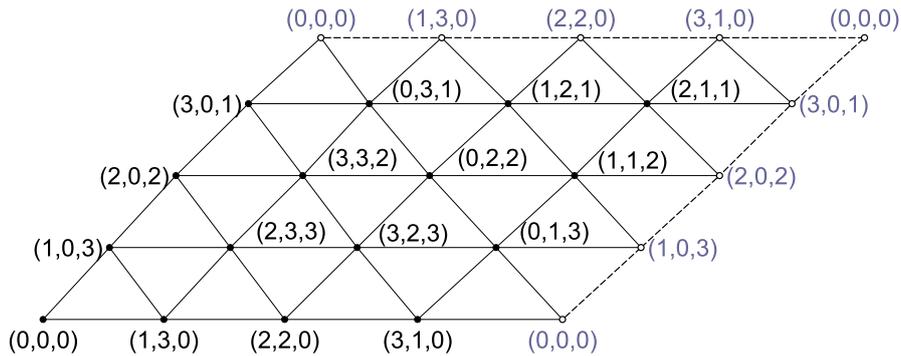


Figura 2.2: Grafo $H_{3,4}$ com repetição de vértices.

O grafo $H_{l,p}$ foi utilizado por Holyer [18] na prova de \mathcal{NP} -completude do problema de partição de arestas. O problema de partição de arestas em K_l 's, chamado de PE_l , considera a partição do conjunto de arestas de um grafo entrada em subconjuntos de arestas tais que cada subconjunto induz um grafo completo de l vértices e temos o seguinte problema de decisão:

PROBLEMA DE PARTIÇÃO DE ARESTAS [18]

ENTRADA: Um grafo e um inteiro $l \geq 3$.

QUESTÃO: Existe uma partição das arestas em k_l 's?

Holyer utilizou o grafo $H_{l,p}$ na demonstração do Teorema 2.3 e provou que existe uma partição do conjunto de arestas do grafo em K_l 's se, e somente se, uma instância do 3SAT é satisfazível.

Teorema 2.3. [18] *O problema de partição de arestas PE_l é \mathcal{NP} -completo para $l \geq 3$.*

Holyer utilizou a mesma redução do problema de PE_l para demonstrar que o problema de partição do conjunto de arestas do grafo em ciclos C_m para $m > 3$, também é \mathcal{NP} -completo.

Corolário 2.4. [18] *O problema de partição de arestas em ciclos C_m é \mathcal{NP} -completo para $m \geq 3$.*

O resultado do problema de partição de arestas em ciclos C_m foi utilizado por Caprara [8] na prova de \mathcal{NP} -completude do problema de *Ordenação por Reversões sem sinal*, onde dados uma permutação π e um inteiro k , decide se é possível com no máximo k reversões, (operação que inverte as posições num intervalo numa permutação), transformar uma permutação em outra.

Como podemos observar, a família $H_{l,p}$ foi muito importante para o estudo da complexidade computacional do problema de Ordenação por Reversões. Agora, mostramos que os grafos da família $H_{l,p}$ são grafos de Cayley e algumas propriedades algébricas. A seguir, definimos o conjunto e a operação para o grupo.

Definição 2.5. *Os vértices do grafo $H_{l,p}$ são os elementos de um conjunto finito $V_{l,p}$.*

Definição 2.6. *A operação $+$ é $(x_{a_1}, \dots, x_{a_l}) + (x_{b_1}, \dots, x_{b_l}) = (x_{a_1} + x_{b_1}, \dots, x_{a_l} + x_{b_l})$, onde x_{a_i} e $x_{b_i} \in (\mathbb{Z}_p, +)$.*

Em seguida mostramos que o par, composto pelo conjunto $V_{l,p}$ e a operação $+$, formam um grupo finito.

Lema 2.7. *$(V_{l,p}, +)$ é um grupo finito.*

Demonstração. Sejam $x_a = (x_{a_1}, \dots, x_{a_l})$, $x_b = (x_{b_1}, \dots, x_{b_l})$, $x_c = (x_{c_1}, \dots, x_{c_l}) \in V_{l,p}$:

1) Verificamos a existência de um elemento identidade. Seja $\iota = (0, 0, \dots, 0) \in V_{l,p}$, observamos que:

$$(0, 0, \dots, 0) + (x_{a_1}, x_{a_2}, \dots, x_{a_l}) = (x_{a_1}, x_{a_2}, \dots, x_{a_l}),$$

$$(x_{a_1}, x_{a_2}, \dots, x_{a_l}) = (x_{a_1}, x_{a_2}, \dots, x_{a_l}) + (0, 0, \dots, 0).$$

De fato ι é o elemento identidade.

2) Exibimos para cada elemento um elemento inverso. Seja $x_a = (x_{a_1}, \dots, x_{a_l}) \in V_{l,p}$, então $-x_a = (-x_{a_1}, \dots, -x_{a_l}) \in V_{l,p}$, tal que cada elemento $x_a + (-x_a) = (-x_a) + x_a = \iota$.

$$\text{Temos que } -x_{a_i} \equiv p - x_{a_i} \pmod{p},$$

$$\text{então } \sum_{i=1}^l -x_{a_i} \equiv -\sum_{i=1}^l x_{a_i} \equiv 0 \pmod{p} \in V_{l,p}.$$

3) Provamos agora que $(V_{l,p}, +)$ é associativo, $(x_a + x_b) + x_c = x_a + (x_b + x_c)$. Aplicando a operação temos:

$$\begin{aligned} & ((x_{a_1}, \dots, x_{a_l}) + (x_{b_1}, \dots, x_{b_l})) + (x_{c_1}, \dots, x_{c_l}) = \\ & (x_{a_1}, \dots, x_{a_l}) + ((x_{b_1}, \dots, x_{b_l}) + (x_{c_1}, \dots, x_{c_l})), \\ & ((x_{a_1} + x_{b_1}, x_{a_2} + x_{b_2}, \dots, x_{a_l} + x_{b_l}) + (x_{c_1}, x_{c_2}, \dots, x_{c_l})) = \\ & (x_{a_1}, x_{a_2}, \dots, x_{a_l}) + ((x_{b_1} + x_{c_1}, x_{b_2} + x_{c_2}, \dots, x_{b_l} + x_{c_l})), \\ & (x_{a_1} + x_{b_1} + x_{c_1}, x_{a_2} + x_{b_2} + x_{c_2}, \dots, x_{a_l} + x_{b_l} + x_{c_l}) = \\ & (x_{a_1} + x_{b_1} + x_{c_1}, x_{a_2} + x_{b_2} + x_{c_2}, \dots, x_{a_l} + x_{b_l} + x_{c_l}). \end{aligned}$$

4) Mostramos que a operação $+$ é fechada em $V_{l,p}$, ou seja, $x_a + x_b \in V_{l,p}$:

$$\text{Como } \sum_{i=1}^l x_{a_i} \equiv 0 \pmod{p}$$

$$\text{e } \sum_{i=1}^l x_{b_i} \equiv 0 \pmod{p},$$

$$\text{então } x_a + x_b = (x_{a_1} + x_{b_1}, x_{a_2} + x_{b_2}, \dots, x_{a_l} + x_{b_l}) \in \mathbb{Z}_p^l,$$

$$\text{ou seja, } \sum_{i=1}^l (x_{a_i} + x_{b_i}) \equiv 0 \pmod{p} \in V_{l,p}.$$

5) Provamos que a cardinalidade de $V_{l,p}$ é finita. Para cada elemento de $V_{l,p}$ temos l coordenadas. Cada coordenada assume qualquer valor entre 0 e $p - 1$, ou seja, exatamente p possibilidades. Mas os elementos de $V_{l,p}$ tem a restrição de que $\sum_{i=1}^l x_{a_i} \equiv 0 \pmod{p}$, portanto, podemos escolher $l - 1$ posições com p possibilidades e a última escolha já está definida com apenas uma possibilidade. Então temos que $|V_{l,p}| = p^{l-1}$. Logo $|V_{l,p}|$ é finita. \square

Lema 2.8. *O grupo $(V_{l,p}, +)$ é abeliano.*

Demonstração. Sejam $x_a = (x_{a_1}, x_{a_2}, \dots, x_{a_l})$ e $x_b = (x_{b_1}, x_{b_2}, \dots, x_{b_l}) \in V_{l,p}$. Mostremos que $(V_{l,p}, +)$ é comutativo, ou seja, $x_a + x_b = x_b + x_a$. Fazemos:

$$\begin{aligned} x_a + x_b &= (x_{a_1}, x_{a_2}, \dots, x_{a_l}) + (x_{b_1}, x_{b_2}, \dots, x_{b_l}), \\ (x_{a_1} + x_{b_1}, x_{a_2} + x_{b_2}, \dots, x_{a_l} + x_{b_l}) &= (x_{b_1} + x_{a_1}, x_{b_2} + x_{a_2}, \dots, x_{b_l} + x_{a_l}), \\ (x_{b_1}, x_{b_2}, \dots, x_{b_l}) + (x_{a_1}, x_{a_2}, \dots, x_{a_l}) &= x_b + x_a. \end{aligned}$$

□

Definição 2.9. *O conjunto gerador $C_{l,p} = \{(c_1, \dots, c_l) \in V_{l,p} : \exists i, j \in \{1, \dots, l\}, i \neq j, \text{ é um conjunto, tal que } c_i = 1, c_j = -1 \text{ e } \forall k \in \{1, \dots, l\} - \{i, j\} \text{ temos que } c_k = 0\}$.*

Por exemplo, quando $V_{3,3} = \{x = (x_1, x_2, x_3) \in \mathbb{Z}_3^3 : \sum_{i=1}^3 x_i \equiv 0 \pmod{p}\}$ temos os seguintes elementos: $V_{3,3} = \{(0, 0, 0), (1, 2, 0), (1, 0, 2), (0, 1, 2), (2, 1, 0), (0, 2, 1), (2, 0, 1), (1, 1, 1), (2, 2, 2)\}$. Para o conjunto $C_{3,3}$ temos os elementos: $C_{3,3} = \{(1, 0, 2), (0, 1, 2), (0, 2, 1), (2, 0, 1), (1, 2, 0), (2, 1, 0)\}$. Neste caso, para o conjunto $C_{3,3}$ ser gerador de $V_{3,3}$, temos que verificar se $V_{3,3}$ é gerado por $C_{3,3}$. Como $C_{3,3} \subseteq V_{3,3}$ temos que conferir apenas os elementos de $V_{3,3}$ que não estão em $C_{3,3}$, ou seja, $\{(0, 0, 0), (1, 1, 1), (2, 2, 2)\}$:

$$(0, 0, 0) = (1, 0, 2) + (2, 0, 1)$$

$$(1, 1, 1) = (1, 0, 2) + (0, 1, 2)$$

$$(2, 2, 2) = (2, 0, 1) + (0, 2, 1)$$

Portanto, o grupo $(V_{3,3}, +)$ é gerado pelo conjunto $C_{3,3}$. Agora, exibimos um conjunto gerador para o grupo $(V_{l,p}, +)$.

Lema 2.10. *$C_{l,p}$ é um conjunto gerador do grupo $(V_{l,p}, +)$.*

Demonstração. Mostramos que existe um caminho entre quaisquer dois elementos deste grupo, ou seja, $(V_{l,p}, +)$ é conexo. Em particular, basta mostrar que existe um caminho entre um elemento qualquer x_a do grupo para o elemento identidade $(0, 0, 0, \dots, 0)$, pois se existe um caminho entre o elemento x_a até o elemento identidade e, além disso, se também existe um caminho entre o elemento x_b até o elemento identidade, logo existe um caminho entre x_a e x_b passando pelo elemento identidade.

Seja $x_a = (x_{a_1}, x_{a_2}, \dots, x_{a_l})$ um elemento qualquer de $V_{l,p}$, onde $\sum_{i=1}^l x_{a_i} \equiv 0 \pmod{p}$ e seja $c_1 = (-1, 0, 0, \dots, 1) \in C_{l,p}$. Então temos:

$$\begin{array}{c}
(x_{a_1}, x_{a_2}, \dots, x_{a_l}) \\
x_{a_1} \text{ parcelas } \left\{ \begin{array}{l} + (-1, 0, 0, \dots, 1) \\ \vdots \\ + (-1, 0, 0, \dots, 1) \end{array} \right. \\
(0, x_{a_2}, \dots, x_{a_1} + x_{a_l}) \\
\\
(0, x_{a_2}, \dots, x_{a_1} + x_{a_l}) \\
x_{a_2} \text{ parcelas } \left\{ \begin{array}{l} + (0, -1, 0, 0, \dots, 1) \\ \vdots \\ + (0, -1, 0, 0, \dots, 1) \end{array} \right. \\
(0, 0, x_{a_3}, \dots, x_{a_1} + x_{a_2} + x_{a_l}) \\
\\
\vdots \\
\\
(0, 0, 0, \dots, x_{a_{l-1}}, x_{a_1} + x_{a_2} + \dots + x_{a_{l-2}} + x_{a_l}) \\
x_{a_{l-1}} \text{ parcelas } \left\{ \begin{array}{l} + (0, 0, 0, 0, \dots, -1, 1) \\ \vdots \\ + (0, 0, 0, 0, \dots, -1, 1) \end{array} \right. \\
(0, 0, 0, \dots, 0, x_{a_1} + x_{a_2} + \dots + x_{a_{l-1}} + x_{a_l})
\end{array}$$

$$(0, 0, 0, \dots, 0, x_{a_1} + x_{a_2} + \dots + x_{a_{l-1}} + x_{a_l}) \equiv 0 \pmod{p} \quad (2.1)$$

Pela Equação 2.1, temos que $x_{a_1} + x_{a_2} + \dots + x_{a_{l-1}} + x_{a_l} \equiv 0 \pmod{p}$. Assim obtemos um caminho de elemento qualquer para o elemento identidade. Portanto, $C_{l,p}$ é um conjunto gerador do grupo $\mathcal{G} = (V_{l,p}, +)$. \square

O conjunto gerador $C_{l,p}$ tem exatamente $l(l-1)$ elementos, ou seja, o valor 1 pode ocorrer em l posições e o valor -1 ocorre nas outras $l-1$ posições. Por exemplo, o conjunto gerador $C_{3,3} = \{(1, 0, 2), (2, 0, 1), (0, 1, 2), (0, 2, 1), (1, 2, 0), (2, 1, 0)\}$ tem 6 elementos. Agora, temos todas as definições e resultados necessários para demonstrarmos que os grafos da família $H_{l,p}$ são grafos de Cayley.

Teorema 2.11. *O grafo $\Gamma(V_{l,p}, E_{l,p})$ é um grafo de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto gerador $C_{l,p}$.*

Demonstração. Pelo conjunto gerador $C_{l,p}$ temos que:

- a. se $c_l = (0, \dots, c_{li}, \dots, c_{lj}, \dots, 0) \in C_{l,p}$ então $-c_l = (0, \dots, c_{lj}, \dots, c_{li}, \dots, 0) \in C_{l,p}$, com $c_{li} = 1$ e $c_{lj} = -1$;
- b. $\iota = (0, 0, 0, \dots, 0) \notin C_{l,p}$.

No Lema 2.10, mostramos que $C_{l,p}$ é o conjunto gerador de $(V_{l,p}, +)$. Assim, temos o grafo $\Gamma(V_{l,p}, E_{l,p})$ definido da seguinte maneira:

- $V = V_{l,p}$;
- $E = \{(x, y)_{c_l} \mid x, y \in V \text{ e } c_l \in C_{l,p}, \text{ tal que, } y = x + c_l\}$.

Com isso, o grafo $\Gamma(V_{l,p}, E_{l,p})$ é conexo, não possui laços, pois o elemento identidade não se encontra no conjunto gerador, nem é orientado, pois o inverso de cada elemento se encontra no conjunto gerador $C_{l,p}$. Portanto, o grafo $\Gamma(V_{l,p}, E_{l,p})$ é um grafo de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto $C_{l,p}$. \square

Definição 2.12. *Dois grafos G e H são isomorfos se existe uma bijeção $f: V(G) \leftrightarrow V(H)$ entre os conjuntos de vértices de G e H , tal que $u, v \in E(G)$ se, e somente se $f(u), f(v) \in E(H)$.*

Corolário 2.13. *O grafo de Cayley $\Gamma(V_{l,p}, E_{l,p})$ é isomorfo ao grafo $H_{l,p}$.*

Demonstração. Pela definição do grafo $H_{l,p}$ e do grupo $(V_{l,p}, +)$, temos os mesmos elementos em ambos casos, então apresentamos a seguinte função: $f: V_{l,p} \rightarrow V_{l,p}$. Mostramos que existe uma aresta entre os vértices u e v no grafo $H_{l,p}$, se, e somente se, existe uma aresta entre u e v no grafo de Cayley $\Gamma(V_{l,p}, E_{l,p})$. Sejam $u = \{u_1, u_2, \dots, u_i, \dots, u_j, \dots, u_l\}$ e $v = \{v_1, v_2, \dots, v_i, \dots, v_j, \dots, v_l\}$ dois vértices quaisquer pertencentes a $V_{l,p}$. Pela definição de aresta do grafo $H_{l,p}$, existe uma aresta entre u e v se $v_i = u_i + 1$ e $v_j = u_j - 1$ ou $v_i = u_i - 1$ e $v_j = u_j + 1$. Agora temos que mostrar que esta aresta pertence ao grafo de Cayley $\Gamma(V_{l,p}, E_{l,p})$, ou seja, existem os elementos c_1 e c_2 em $C_{l,p}$, tal que $v = u + c_1$ ou $u = v + c_2$. Sejam $c_1 = \{0, 0, \dots, 1, \dots, -1, \dots, 0\} \in C_{l,p}$ e $c_2 = \{0, 0, \dots, -1, \dots, 1, \dots, 0\} \in C_{l,p}$ estes elementos. Logo, temos a aresta no grafo de Cayley $H_{l,p}$ e os grafos são isomorfos. \square

Vimos, nesta seção, que o grupo $(V_{l,p}, +)$ é finito e abeliano. Também mostramos que os grafos da família $H_{l,p}$ são grafos de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto gerador $C_{l,p}$, e portanto, são grafos vértice-transitivos.

2.2 Grafo $H'_{l,p}$

Nesta seção, apresentamos o grafo $H'_{l,p}$. O grafo $H'_{l,p}$ é um subgrafo gerador do grafo $H_{l,p}$, obtido pela remoção de alguns elementos do conjunto gerador $C_{l,p}$, com isso temos um grafo mais esparsa e com um conjunto gerador minimal. Encontrar um ciclo hamiltoniano em um grafo de Cayley com um conjunto gerador minimal é mais difícil, pois temos menos caminhos entre os pares de vértices do grafo para escolher.

Definição 2.14. *O conjunto gerador $C'_{l,p} = \{(c'_1, \dots, c'_l) \in V_{l,p} : \exists i \in \{1, \dots, l-1\}, \text{ tal que } c'_i = 1 \text{ e } c'_l = -1, \text{ ou } c'_i = -1 \text{ e } c'_l = 1, \text{ e } \forall k \in \{1, \dots, l-1\} - \{i\}, \text{ temos que } c'_k = 0\}$*

O conjunto gerador $C'_{l,p}$ consiste em exatamente $2(l-1)$ elementos, sendo que, o valor 1 ocorre nas $l-1$ posições e o valor -1 também ocorre nas $l-1$ posições. Por exemplo, o conjunto gerador $C'_{3,3} = \{(1, 0, 2), (2, 0, 1), (0, 1, 2), (0, 2, 1)\}$ consiste em 4 elementos.

Lema 2.15. *$C'_{l,p}$ é o conjunto gerador do grupo $(V_{l,p}, +)$.*

Demonstração. Análogo ao visto na demonstração do Lema 2.10. □

Teorema 2.16. *O grafo $H'_{l,p}(V, E')$ é um grafo de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto gerador $C'_{l,p}$.*

Demonstração. Pelo conjunto gerador $C'_{l,p}$, temos que:

- a. se $c'_l = (0, \dots, c'_{li}, \dots, 0, \dots, c'_{lj}) \in C'_{l,p}$, então $-c'_l = (0, \dots, c'_{lj}, \dots, 0, \dots, c'_{li}) \in C'_{l,p}$, com $c'_{li} = 1$ e $c'_{lj} = -1$;
- b. $\iota = (0, 0, 0, \dots, 0) \notin C'_{l,p}$.

No Lema 2.10, mostramos que $C'_{l,p}$ é o conjunto gerador de $(V_{l,p}, +)$. Assim, temos o grafo $H'_{l,p}$ definido da seguinte forma:

- $V = V_{l,p}$;
- $E' = \{(x, y)_{c'_l} \mid x, y \in V \text{ e } c'_l \in C'_{l,p}, \text{ tal que, } y = x + c'_l\}$.

Com isso, o grafo $H'_{l,p}$ é conexo, não possui laços, pois o elemento identidade não se encontra no conjunto gerador, nem é orientado, pois o inverso de cada elemento se encontra no conjunto gerador $C'_{l,p}$. Portanto, o grafo $H'_{l,p}$ é um grafo de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto $C'_{l,p}$. □

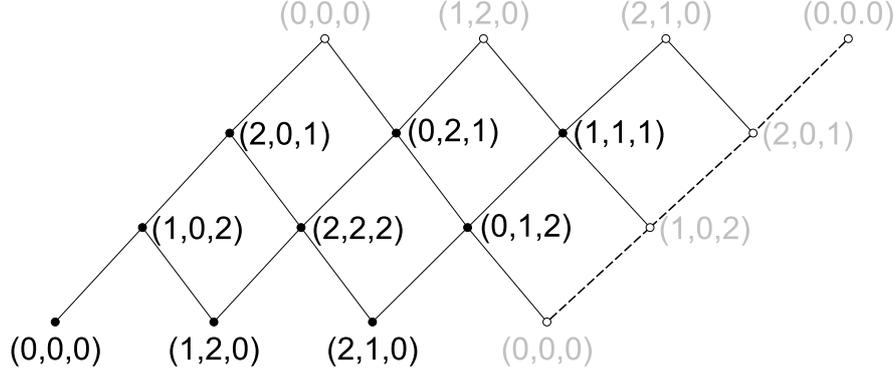


Figura 2.3: Grafo $H'_{3,3}$ com repetição de vértices.

Por exemplo, o conjunto $C'_{3,3}$ consiste nos elementos: $\{(1, 0, 2), (0, 1, 2), (0, 2, 1), (2, 0, 1)\}$. Os elementos $(0, 0, 0)$, $(1, 1, 1)$, $(2, 2, 2)$, $(1, 2, 0)$, $(2, 1, 0)$ são gerados por: $(0, 0, 0) = (1, 0, 2) + (2, 0, 1)$; $(1, 1, 1) = (1, 0, 2) + (0, 1, 2)$, $(2, 2, 2) = (2, 0, 1) + (0, 2, 1)$, $(1, 2, 0) = (1, 0, 2) + (0, 2, 1)$ e $(2, 1, 0) = (2, 0, 1) + (0, 1, 2)$. Assim, o grupo $(V_{3,3}, +)$ é gerado pelo conjunto $C'_{3,3}$ e obtemos o grafo de Cayley $H'_{3,3}$.

Na Figura 2.3, apresentamos uma representação do grafo $H'_{3,3}$ com repetição de vértices. Observamos pelas Figuras 2.3 e 2.1 dos grafos $H'_{3,3}$ e $H_{3,3}$, que o grafo $H'_{3,3}$ é um subgrafo esparsado do grafo $H_{3,3}$. O grafo $H'_{3,3}$ tem menos arestas do que o grafo $H_{3,3}$. O grafo $H_{3,3}$ tem grau 6 e o grafo $H'_{3,3}$ tem grau 4.

Para $l \geq 3$ e $p \geq 3$, definimos o grafo $H'_{l,p} = (V_{l,p}, E'_{l,p})$ como:

- $V_{l,p} = \{x = (x_1, \dots, x_l) \in \mathbb{Z}_p^l : \sum_{i=1}^l x_i \equiv 0 \pmod{p}\}$;
- $E'_{l,p} = \{xy : \text{existem } i, j \text{ que satisfazem } y_k \equiv_p x_k \text{ para todo } k \neq \{i, j\} \text{ e } (y_i \equiv_p x_i + 1, y_j \equiv_p x_j - 1 \text{ ou } y_i \equiv_p x_i - 1, y_j \equiv_p x_j + 1) \text{ onde } j = l\}$.

Lema 2.17. *O grafo $H'_{l,p}$ tem as seguintes propriedades:*

- a. o número total de vértices é $n = p^{l-1}$;
- b. o grau de cada vértice é $2(l-1)$;
- c. o número total de arestas é $(l-1)p^{l-1}$.
- d. se $p > 4$, então o grau é menor que $\log_2 n$.

Demonstração. O conjunto de vértices é o mesmo do grafo $H_{l,p}$, portanto a quantidade de vértices do grafo $H'_{l,p}$ é dada pelo fato de que em $l-1$ coordenadas do vértice podemos variar o valor da coordenada entre 0 e $p-1$ e pela restrição da definição dos vértices do grafo $H'_{l,p}$ temos somente um valor para o último coordenada. Portanto $|V| = p^{l-1}$. Pela definição do conjunto de arestas, observamos que o valor

1 pode ser colocado nas primeiras $l - 1$ posições do vértice, ou também podemos colocar o valor -1 nestas $l - 1$ posições. Logo o grau de cada vértice é $2(l - 1)$. $2(l - 1) < \log_2 p^{l-1}$, $2(l - 1) < (l - 1) \log_2 p$, $2 < \log_2 p$, $p > 4$. \square

Vimos, nesta seção, que os grafos da família $H'_{l,p}$ são grafos de Cayley do grupo $(V_{l,p}, +)$ associado ao conjunto gerador $C'_{l,p}$, e portanto, são grafos vértice-transitivos.

2.3 Ciclo hamiltoniano

2.3.1 Grafos de Cayley abeliano

Nesta seção, apresentamos o resultado de Marušič sobre ciclo hamiltoniano em um grafo de Cayley associado a um grupo abeliano [29]. Em seguida, mostramos que os grafos das famílias $H_{l,p}$ e $H'_{l,p}$ admitem ciclos hamiltonianos. A seguir, apresentamos as definições e notações usadas por Marušič.

Definição 2.18. *Sejam \mathcal{G} um grupo e ι o elemento identidade. Dado $g \in G$, dizemos que j é a ordem de g , denotado por $|g| = j$, se j é o menor inteiro positivo, tal que $g^j = \iota$.*

Dado M um subconjunto de \mathcal{G} , temos as seguintes definições: $M^{-1} = \{x^{-1} : x \in M\}$; $M_0 = M - \{\iota\}$; $M^* = M_0 \cup M_0^{-1}$; $\langle M \rangle$ é o subconjunto de \mathcal{G} gerado por M ; Se $\langle M \rangle = \mathcal{G}$, então M é chamado de conjunto gerador de \mathcal{G} .

Uma *sequência* em \mathcal{G} é uma sequência onde todos os seus termos são elementos de \mathcal{G} . A sequência sem termos é denotada por \emptyset . Dada as sequências $S = [s_1, s_2, \dots, s_r]$ e $T = [t_1, t_2, \dots, t_q]$, onde $s_i \in G$ para $1 \leq i \leq r$ e $t_k \in G$ para $1 \leq k \leq q$, o produto ST é definido por $[s_1, s_2, \dots, s_r, t_1, t_2, \dots, t_q]$. O produto parcial $\Pi_i(S)$ de S é formado pela operação do grupo $s_1 s_2 \dots s_i$.

Seja $S = [s_1, s_2, \dots, s_r]$ uma sequência de \mathcal{G} , então dizemos que S é uma *sequência hamiltoniana* se: $r = |\mathcal{G}|$; $\Pi_r(S) = \iota$; $\Pi_i(S) \neq \Pi_j(S)$ se $i \neq j$ para $1 \leq i, j \leq r$.

Se $s_i \in M$, para $i = 1, 2, \dots, r$, então a sequência S é chamada de M -sequência, e se S é hamiltoniana, então S é chamada de M^* -sequência. O conjunto $\mathcal{H}(M, \mathcal{G})$ é formado por todas as M^* -sequências de \mathcal{G} .

Lema 2.19. [29] *O grafo de Cayley (\mathcal{G}, M) admite ciclo hamiltoniano se, e somente se, $\mathcal{H}(M, \mathcal{G}) \neq \emptyset$.*

Lema 2.20. [29] *Seja M um conjunto gerador de um grupo abeliano \mathcal{G} e M' é um subconjunto não vazio de M_0 . Se $S, T \in \mathcal{H}(M', \langle M' \rangle)$ e $l_s = l_r$, então existe uma sequência Q em \mathcal{G} , tal que $\bar{S}Q, \bar{T}Q \in \mathcal{H}(M, \mathcal{G})$. Onde \bar{S} é a sequência S sem o último elemento.*

Demonstração. Provamos por indução na cardinalidade de $M_0 \setminus M'$. Se $M_0 \setminus M' = \emptyset$ então o Lema 2.20 é verdadeiro com $Q = \emptyset$. Seja $M_0 \setminus M' \neq \emptyset$, $g \in M_0 \setminus M'$, $H = \langle M \setminus \{g\} \rangle$ e j sendo o menor inteiro positivo onde $g^j \in H$. Pela hipótese de indução existe uma sequência R em H , tal que $\bar{S}R, \bar{T}R \in \mathcal{H}(M \setminus \{g\}, H)$. Se $W = \bar{S}R$, seja Q a sequência $\bar{R}(W, [g]^j)[l_w][g^{-1}]^{j-1}$, se j é ímpar, e seja a sequência $\bar{R}(W, [g]^j)[g](\bar{W})^{-1}[g^{-1}]^{j-1}$, se j é par. Neste caso, temos que $\bar{S}Q, \bar{T}Q \in \mathcal{H}(M, G)$. \square

Teorema 2.21. [29] *Todo grafo de Cayley conexo de um grupo abeliano e com ordem maior ou igual a três admite ciclo hamiltoniano.*

Demonstração. Pelo Lema 2.19 é suficiente mostrar que se M é um conjunto gerador de um grupo abeliano \mathcal{G} de ordem no mínimo três, então $\mathcal{H}(M, G) \neq \emptyset$. Se M contém um elemento x de ordem $n \geq 3$, então basta tomar $M' = \{x\}$ e $S = [x]^n$. Se M não contém elementos de ordem no mínimo 3, então existem (uma vez que \mathcal{G} tem ordem no mínimo três) pelo menos dois elementos y e z de ordem dois, tal que \mathcal{G} tenha ordem no mínimo três. Neste caso, basta escolher $M' = \{y, z\}$ e $S = ([y][z])^2$. Segue pelo Lema 2.20 que $\mathcal{H}(M, G) \neq \emptyset$. \square

Sabemos que os grafos das famílias $H_{l,p}$ e $H'_{l,p}$ são grafos de Cayley associados a grupos abelianos. Assim, obtemos o seguinte resultado.

Corolário 2.22. *Os grafos das famílias $H_{l,p}$ e $H'_{l,p}$ admitem ciclos hamiltonianos.*

Demonstração. Pelo Lema 2.8, temos que o grupo $(V_{l,p}, +)$ é abeliano e pelos resultados dos Teoremas 2.11 e 2.16, temos que os grafos das famílias $H_{l,p}$ e $H'_{l,p}$ são grafos de Cayley. \square

Usando o resultado de Marušič [29], podemos construir um algoritmo que retorna ciclos hamiltonianos nos grafos $H_{l,p}$ e $H'_{l,p}$. Pelo Lema 2.20, sabemos que existem duas sequências $\bar{S}Q$ e $\bar{T}Q$ no grafo $H'_{l,p}$, pois ele é um grafo de Cayley associado a um grupo abeliano. Pelo Teorema 2.21 temos um processo recursivo para construir a sequência $\bar{S}Q$. Depois, utilizamos esta sequência em qualquer vértice do grafo $H'_{l,p}$ para obter um ciclo hamiltoniano no grafo. Esta operação possui a complexidade de $O(p^{l-1})$, pois o grafo $H'_{l,p}$ possui p^{l-1} vértices e o ciclo hamiltoniano é obtido através da sequência $\bar{S}Q$ que tem mesma cardinalidade dos vértices do grafo. Como $E(H'_{l,p}) \subset E(H_{l,p})$, temos que este ciclo também é um ciclo no grafo $H_{l,p}$.

Por exemplo, usando os resultados acima no grafo $H'_{3,4}$ obtemos a seguinte sequência hamiltoniana $\bar{S}Q = \{(1, 0, 3), (1, 0, 3), (1, 0, 3), (0, 1, 3), (3, 0, 1), (3, 0, 1), (0, 1, 3), (1, 0, 3), (1, 0, 3), (0, 1, 3), (3, 0, 1), (3, 0, 1), (3, 0, 1), (0, 3, 1), (0, 3, 1), (0, 3, 1)\}$. Aplicando esta sequência no vértice $(0, 0, 0)$ temos a sequência $C = \{(1, 0, 3), (2, 0, 2), (3, 0, 1), (3, 1, 0), (2, 1, 1), (1, 1, 2), (1, 2, 1), (2, 2, 0), (3, 2, 3), (3, 3, 2), (2, 3, 3),$

$(1, 3, 0), (0, 3, 1), (0, 2, 2), (0, 1, 3), (0, 0, 0)\}$, que é um ciclo hamiltoniano nos grafos $H'_{3,4}$ e $H_{3,4}$.

Nesta seção, tratamos do resultado de Marušič sobre ciclo hamiltoniano em grafos de Cayley associado a um grupo abeliano para definir um ciclo hamiltoniano nos grafos das famílias $H_{l,p}$ e $H'_{l,p}$ com complexidade de tempo e espaço de $O(p^{l-1})$. Na próxima seção, utilizamos um código Gray para apresentar outro ciclo hamiltoniano nos grafos das famílias $H_{l,p}$ e $H'_{l,p}$.

2.3.2 Ciclo hamiltoniano com código Gray

Nesta seção, apresentamos outra construção de um ciclo hamiltoniano nos grafos $H_{l,p}$ e $H'_{l,p}$ com a utilização do código (p, l) -Gray. A ideia principal é a criação de um grafo G de um código $(p, l - 1)$ -Gray com $l - 1$ coordenadas na base p e a demonstração de que o grafo G é isomorfo a um ciclo no grafo $H'_{l,p}$.

Os códigos binários refletidos também são conhecidos como códigos Gray. O código Gray foi proposto por Frank Gray [16]. O termo código Gray é utilizado para referir qualquer código, tal que palavras adjacentes do código diferem apenas por um dígito e uma posição. Alguns códigos Gray são cíclicos, ou seja, o último elemento difere apenas de um dígito do primeiro elemento.

Algoritmo 2.1: (p, l) -código Gray [22, 40]

Entrada: *valorBase10*, l , p

Saída: *valorGray*

```

1 início
2    $t = \text{valorBase10}$ 
3   para  $i = l - 1$  até 0 faça
4      $\text{baseP}[i] = t \bmod p$ 
5      $t = t/p$ 
6    $t = 0$ 
7   para  $i = 0$  até  $l - 1$  faça
8      $\text{valorGray}[i] = (\text{baseP}[i] - t) \bmod p$ 
9      $t = t + (\text{valorGray}[i] - p)$ 
10 fim
```

O Algoritmo 2.1 transforma um valor decimal qualquer em um código (p, l) -Gray, cada elemento do código é formado por l dígitos na base p . Por exemplo, a sequência dos elementos no código $(4, 2)$ -Gray é $G = \{(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (1, 0), (1, 1), (1, 2), (2, 2), (2, 3), (2, 0), (2, 1), (3, 1), (3, 2), (3, 3), (3, 0)\}$. Este código Gray G é isomorfo ao ciclo $C = \{(0, 0, 0), (0, 1, 3), (0, 2, 2), (0, 3, 1), (1, 3, 0), (1, 0, 3), (1, 1, 2),$

$(1, 2, 1), (2, 2, 0), (2, 3, 3), (2, 0, 2), (2, 1, 1), (3, 1, 0), (3, 2, 3), (3, 3, 2), (3, 0, 1)\}$. O ciclo C é isomorfo a um ciclo no grafo $H'_{3,4}$.

Primeiramente, mostramos que o último elemento do código (p, l) -Gray possui apenas um dígito de diferença do primeiro elemento, assim temos um código Gray cíclico e podemos utilizar este ciclo para encontrar o ciclo hamiltoniano no grafo $H'_{l,p}$.

Teorema 2.23. *O código (p, l) -Gray do Algoritmo 2.1 é cíclico.*

Demonstração. Consideramos o primeiro elemento sendo o elemento identidade e valorGray o último número de l dígitos e base p do código (p, l) -Gray. Seja BaseP = $(p - 1, p - 1, p - 1, \dots, p - 1) \in \mathbb{Z}_p^l$ o último elemento na sequência de base p . Agora aplicamos o Algoritmo 2.1 em BaseP para obtermos valorGray no código (p, l) -Gray.

$t = 0$ <p>Para o primeiro dígito, temos:</p> $\text{valorGray}[0] = (\text{BaseP}[0] - t) \pmod p$ $\text{valorGray}[0] = (p - 1 - 0) \pmod p$ $\text{valorGray}[0] = p - 1$ $t+ = \text{valorGray}[0] - p$ $t+ = p - 1 - p$ $t = -1$ <p>Para o segundo dígito, temos:</p> $\text{valorGray}[1] = (\text{BaseP}[1] - t) \pmod p$ $\text{valorGray}[1] = (p - 1 - (-1)) \pmod p$ $\text{valorGray}[1] = (p) \pmod p$ $\text{valorGray}[1] = 0$ $t+ = \text{valorGray}[1] - p$ $t+ = 0 - p$ $t = -p - 1$	<p>Para o terceiro dígito, temos:</p> $\text{valorGray}[2] = (\text{BaseP}[2] - t) \pmod p$ $\text{valorGray}[2] = (p - 1 - (-p - 1)) \pmod p$ $\text{valorGray}[2] = (2p) \pmod p$ $\text{valorGray}[2] = 0$ $t+ = \text{valorGray}[2] - p$ $t+ = 0 - p$ $t = -2p - 1$ \vdots $t = -(l - 2)p - 1$ <p>Para o último dígito, temos:</p> $\text{valorGray}[l-1] = (\text{BaseP}[l-1] - t) \pmod p$ $\text{valorGray}[l-1] = (p - 1 - (-(l - 2)p - 1)) \pmod p$ $\text{valorGray}[l - 1] = ((l - 1)p) \pmod p$ $\text{valorGray}[l - 1] = 0$ $t+ = \text{valorGray}[l - 1] - p$ $t+ = 0 - p$ $t = -(l - 1)p - 1$
---	---

Como $\text{valorGray} = (p - 1, 0, 0, 0, \dots, 0)$ temos que verificar se valorGray difere em apenas um dígito do elemento identidade. Fazendo $\text{valorGray} + (1, 0, 0, 0, \dots, 0)$, obtemos $(0, 0, 0, 0, \dots, 0)$. Portanto, o código (p, l) -Gray forma um ciclo. \square

Definimos um ciclo C' através do código $(p, l - 1)$ -Gray utilizando o intervalo decimal entre 0 e $p^{l-1} - 1$ e obtemos um intervalo entre 0 e $p^{l-1} - 1$ em código Gray. Logo, o ciclo C' tem exatamente p^{l-1} vértices e cada vértice tem $l - 1$ coordenadas. Pela definição do código Gray, temos que cada vértice tem diferença de apenas um dígito entre seus vizinhos. Seja $c' = \{c'_1, c'_2, \dots, c'_{l-1}\} \in V(C')$, definimos um ciclo C formado pelo vértice $c = \{c_1, c_2, \dots, c_{l-1}, c_l\} \in V(C)$, onde $c_i = c'_i$ para valores entre 1 e $l - 1$ e $c_l = (p - (\sum_{i=1}^{l-1} c_i \bmod p))$, tal que $V(C) = \{c = (c_1, c_2, \dots, c_l) \in \mathbb{Z}_p^l : \sum_{i=1}^l c_i \equiv 0 \pmod p\}$.

Teorema 2.24. *A partir do ciclo C obtido pelo código $(p, l - 1)$ -Gray, podemos obter um ciclo hamiltoniano nos grafos $H_{l,p}$ e $H'_{l,p}$.*

Demonstração. Pela definição, temos que os vértices do ciclo C são os mesmos vértices do grafo $H'_{l,p}$. Agora, mostramos que $E(C) \in E(H'_{l,p})$. Seja $e = (u, v) \in E(C)$ uma aresta qualquer do ciclo C , com o vértice $u = \{u_1, u_2, \dots, u_{l-1}, u_l\}$ e o vértice $v = \{v_1, v_2, \dots, v_{l-1}, v_l\}$. Sejam os vértices $u' = \{u'_1, u'_2, \dots, u'_{l-1}\}$ e $v' = \{v'_1, v'_2, \dots, v'_{l-1}\}$, pela definição dos vértices do ciclo C , temos uma aresta $e' = (u', v') = e \in E(C)$. Pela definição do código Gray, temos em um coordenada a diferença de um dígito de u' para v' onde $u_i = v_i + 1$. Podemos afirmar sem perda de generalidade que $\sum_i^{l-1} v'_i \bmod p = \sum_i^{l-1} u'_i \bmod p + 1$. Consideramos os coordenadas $u_l = p - (\sum_i^{l-1} u'_i \bmod p)$ e $v_l = p - (\sum_i^{l-1} v'_i \bmod p)$. Fazendo $v_l = p - (\sum_i^{l-1} u'_i \bmod p) + 1$, temos que $v_l = u_l + 1$. Logo, temos que $u_i = v_i + 1$ e $u_l = v_l - 1$, portanto de acordo com a definição de aresta do grafo $H'_{l,p}$, temos que $e \in E(H'_{l,p})$. Como $E(H'_{l,p}) \subset E(H_{l,p})$, obtemos um ciclo nos grafos $H_{l,p}$ e $H'_{l,p}$. \square

Podemos implementar um algoritmo de complexidade de tempo $O(p^{l-1})$ que encontra a sequência dos vértices que formam um ciclo hamiltoniano no grafo $H_{l,p}$. Para isso, utilizamos os valores decimais entre 0 e p^{l-1} e, pelo Algoritmo 2.1, obtemos os valores no código (p, l) -Gray. Como o Algoritmo 2.1 executa em tempo constante de l dígitos para cada código, então a complexidade de tempo do algoritmo é $O(lp^{l-1})$ e de espaço é $O(l)$, ou seja, temos um algoritmo melhor que o apresentado por Marušič [29].

2.4 Diâmetro

Nesta seção, estudamos o diâmetro dos grafos das famílias $H_{l,p}$ e $H'_{l,p}$ e apresentamos alguns resultados sobre distância nestas famílias. A distância $d(u, v)$ entre os vértices

u e v é o comprimento do menor caminho entre u e v . O diâmetro, denotado por D , é a maior distância entre todos os pares de vértices.

Pela Proposição 1.5, temos que os grafos de Cayley são vértice-transitivos, então podemos sempre ver a distância entre quaisquer dois vértices arbitrários como a distância entre o vértice de origem p e vértice identidade, simplesmente renomeando adequadamente os elementos que representam os vértices do grafo.

Por exemplo, seja $\bar{c}ab$ o vértice de origem e $bc\bar{a}$ o vértice de destino. Podemos mapear o vértice de destino para o vértice identidade abc renomeando os elementos do vértice como $b \rightarrow a$, $b \rightarrow \bar{a}$, $c \rightarrow b$, $\bar{c} \rightarrow b$, $\bar{a} \rightarrow c$ e $a \rightarrow c$. Com esse mapeamento o vértice de origem torna-se $bc\bar{a}$. Então, os caminhos entre o vértice de origem e o vértice de destino tornar-se isomorfo ao caminho entre o vértice $bc\bar{a}$ e o vértice identidade abc no grafo renomeado.

Assim, a nossa discussão subsequente sobre o caminho de um vértice de origem para um vértice de destino, o vértice de destino é sempre assumido como sendo o vértice identidade ι sem qualquer perda de generalidade. Portanto, definimos $d(p)$ como o comprimento de uma sequência mínima que gera o elemento p [44]. Portanto, para encontrar o diâmetro é suficiente calcular a maior distância entre o elemento identidade e todos os outros vértices. Os vértices que possuem a maior distância são chamados de diametrais.

2.4.1 Grafo $H_{l,p}$

Para encontrar o diâmetro do grafo $H_{l,p}$, é suficiente obtermos o caminho mínimo que gera todos os vértices do grafo $H_{l,p}$ em relação ao elemento identidade. Assim, se temos as distâncias dos vértices em relação a identidade, então temos o diâmetro do grafo $H_{l,p}$, que é a maior distância que foi encontrada. Antes de encontrar a distância do vértice para o elemento identidade precisamos do seguinte resultado.

Lema 2.25. *Seja x' uma permutação das coordenadas de um vértice x do grafo $H_{l,p}$, tal que as coordenadas em x' estão na ordem crescente, onde $x'_i \leq x'_{i+1}$, temos que a distância $d(x) = d(x')$.*

Demonstração. Provamos por indução no valor de r , que representa a quantidade de trocas realizadas nos coordenadas do vértice x' , para obter o vértice x .

Caso Base: Para $r = 1$, seja $x = \{x_1, x_2\}$ um vértice do grafo $H_{l,p}$. Consideramos sem perda de generalidade que $x_2 \leq x_1$. Dado $x' = \{x'_1, x'_2\}$ um vértice onde $x'_1 = x_2$ e $x'_2 = x_1$, ou seja, $x'_i \leq x'_{i+1}$. Provamos que $d(x) = d(x')$.

Vamos subtrair x'_1 unidades para obter zero na primeira posição, assim precisamos somar x'_1 unidades para obter zero na segunda posição, pois $x'_2 + x'_1 \equiv_p 0$. Portanto, a distância de $d(x') = x'_1$. Sem perda de generalidade, podemos fazer as

mesmas operações com o vértice x . Mas $x_2 \leq x_1$, assim é necessário subtrair x_2 unidades para obter zero na segunda posição de x e somar x_2 unidades para obter zeros na primeira posição de x , então $d(x) = x_2$. Como $x_2 = x'_1$ temos que $d(x') = d(x)$ para $l = 2$.

Hipótese de indução: A distância de $d(y) = d(y')$ para $r \geq 2$ até $r - 1$ trocas.

Passo indutivo: Seja x um vértice do grafo $H_{l,p}$ e x' um vértice onde $x'_i \leq x'_{i+1}$. Seja y um vértice com $r - 1$ trocas, ou seja, escolhamos duas posições em x onde $x_1 \geq x_2$ e trocando x_1 por x_2 , obtemos o vértice y' . Pela hipótese de indução, a distância $d(y) = d(y')$. Considere que x' está a 1 troca de y' , pelo caso base temos que $d(x') = d(y')$, assim $d(y') = d(y)$. Como y também está a 1 troca de x temos que $d(y) = d(x)$ e, portanto $d(x) = d(x')$. \square

Sabemos pelo Lema 2.25 que quaisquer dois vértices x e x' possuem a mesma distância, tal que estes vértices tenham os mesmos elementos que podem estar em ordens diferentes. Assim, podemos gerar qualquer um destes vértices a partir da identidade, e temos assim a sua distância.

Pelo Lema 2.26, podemos calcular a distância de um vértice do grafo $H_{l,p}$, dividindo ele em duas partes. A primeira parte do vértice contém l_1 coordenadas e estas coordenadas são subtraídas de 1 unidade até que os l_1 coordenadas atinjam 0. A segunda parte do vértice contém l_2 coordenadas e estas coordenadas são adicionados de 1 unidade até que os l_2 coordenadas atinjam p . Seja s_{l_1} a soma dos l_1 coordenadas da primeira parte. Seja s_{l_2} a soma de $p - x'_i$ dos l_2 coordenadas da segunda parte. Pela definição temos que $s_{l_1} - s_{l_2} = 0$. Denotamos por $s_t = \sum_{i=1}^l x_i$.

Lema 2.26. A distância do vértice x do grafo $H_{l,p}$ é $d(x) = s_{l_1}$, onde $s_{l_1} = \sum_{i=1}^{l_1} x'_i =$

$$\sum_{j=l_1+1}^l (p - x'_j) \text{ e } l_1 = l - \frac{1}{p} \sum_{i=1}^l x'_i.$$

Demonstração. Consideramos $x = \{x_1, x_2, \dots, x_l\}$ um vértice do grafo $H_{l,p}$ e seja o vértice x' obtido pela ordenação crescente das coordenadas de x . Pelo Lema 2.25, temos que $d(x) = d(x')$.

Dados $0 < l_1 < l$ e $l_2 = l - l_1$, tem-se que $l_1 = l - l_2$. Seja $s_{l_1} = \sum_{i=1}^{l_1} x'_i$ o número de operações de subtração que temos que realizar com os coordenadas de l_1 e $s_{l_2} = \sum_{j=|l_1|+1}^{|l|} (p - x'_j)$ o número de operações de soma com os coordenadas de l_2 .

Seja $l_1 = 1$, com isso $s_{l_1} = x'_1$ e $s_{l_2} = \sum_{j=2}^l (p - x'_j)$. Se $s_{l_1} \neq s_{l_2}$, então fazemos $l_1 = 2$, $s_{l_1} = s_{l_1} + x'_2$ e $s_{l_2} = s_{l_2} - (p - x'_2)$. Temos que fazer isso até que $s_{l_1} = s_{l_2}$,

logo temos a quantidade de operações necessárias para zerar todas as posições do vértice x' . Portanto, a distância é $d(x') = s_{l_1}$ quando $s_{l_1} = s_{l_2}$. \square

Usamos o resultado do Lema 2.26 para apresentar o Algoritmo 2.2 que encontra a distância de um vértice x para o elemento identidade ι do grafo $H_{l,p}$ com complexidade $O(lp \log l)$, sem o uso de memória extra.

Algoritmo 2.2: Distância do grafo $H_{l,p}$

Entrada: l, p , vértice
Saída: $d(\text{vértice})$

```

1 início
2   Ordene(vértice)                                % tal que  $x'_i \leq x'_{i+1}$  para  $1 \leq i \leq l$ 
3   para  $i = 1$  até  $l$  faça
4      $sl_2 = (p - \text{vértice}[i]) + sl_2$ 
5      $sl_1 = \text{vértice}[0]$ 
6     enquanto  $sl_1 \neq sl_2$  faça
7        $i = i + 1$ 
8        $sl_1 = sl_1 + \text{vértice}[i]$ 
9        $sl_2 = sl_2 - (p - \text{vértice}[i])$ 
10     $d(\text{vértice}) = sl_1$ 
11 fim
```

A partir dos Algoritmos 2.1 e 2.2, apresentamos o Algoritmo 2.3 para encontrar o diâmetro do grafo $H_{l,p}$. A complexidade do Algoritmo 2.3 é $O(n)$, já que o grafo possui $n = p^{l-1}$ vértices, temos para o algoritmo complexidade exponencial em relação aos parâmetros l e p , que definem o grafo, pois a quantidade de vértices também é exponencial em relação a estes parâmetros. O Algoritmo 2.3 encontra a distância de cada vértice em relação ao elemento identidade ι , depois retorna o maior valor encontrado.

Algoritmo 2.3: Diâmetro do grafo $H_{l,p}$

Entrada: l, p
Saída: D

```

1 início
2    $t = p^{l-1} - 1$ 
3    $D = 0$ 
4   para  $j = 1$  até  $t$  faça
5     vértice = (p,l)-Código Gray(j,l,p)           % Algoritmo 2.1
6      $d = \text{distância}(l, p, \text{vértice})$            % Algoritmo 2.2
7     se  $d > D$  então
8        $D = d$ 
9   retorna  $D$ 
10 fim
```

Na Tabela 2.1, mostramos os diâmetros do grafo $H_{l,p}$ para valores $2 \leq l \leq 10$ e $2 \leq p \leq 10$. Para preencher a tabela, utilizamos o Algoritmo 2.3 em um computador com processador *i7* da intel. A computação levou aproximadamente $24h$ para efetuar o cálculo dos diâmetros da Tabela 2.1.

Tabela 2.1: Diâmetros do grafo $H_{l,p}$, para valores de l e p entre 2 e 10.

l/p	2	3	4	5	6	7	8	9	10
2	1	1	2	2	3	3	4	4	5
3	1	2	2	3	4	4	5	6	6
4	2	2	4	4	6	6	8	8	10
5	2	3	4	6	6	8	9	10	12
6	3	4	6	6	9	9	12	12	15
7	3	4	6	8	9	12	12	15	16
8	4	5	8	9	12	12	16	16	20
9	4	6	8	10	12	15	16	20	20
10	5	6	10	12	15	16	20	20	25

Apesar da complexidade linear em relação ao número de vértices, temos o problema de que o tamanho do grafo $H_{l,p}$ é exponencial, em relação aos parâmetros l e p . Com isso temos um algoritmo ineficiente para o cálculo do diâmetro.

Precisamos de um algoritmo que encontre o diâmetro sem fazer a análise da distância dos vértices do grafo $H_{l,p}$. Podemos fazer isto encontrando um vértice diametral, ou seja, aquele cuja distância é a maior. Analisando o Algoritmo 2.2 chegamos as seguintes conclusões: Sabemos que $l_1 = l - s_t/p$. Logo, temos que $l_2 = s_t/p$. Como a distância é um número inteiro, existe um número $q \in \mathbb{Z}$, tal que $s_t = p \cdot q$. Por exemplo, para calcular a distância do vértice $(4, 4, 4, 4)$ do grafo $H_{4,8}$ fazemos $l_1 = 4 - 16/8$, $l_1 = 2$ e $d = \sum_{i=1}^{l_1} x'_i$, $d = 4 + 4 = 8$.

Apesar de sabermos o valor da soma das coordenadas, que é $s_t = p(l - l_1)$, não sabemos o melhor valor para l_1 em função de l e p , ou seja, não temos uma relação direta entre s_t e s_{l_1} , dificultando assim a computação do diâmetro. Por exemplo, o grafo $H_{3,6}$ tem como vértice diametral $v = (2, 2, 2)$, ou seja, o valor de l_1 é $6 = 6(3 - l_1)$, $l_1 = 2$. Assim, sabendo o melhor valor para l_1 , temos o diâmetro, que neste caso é 4.

Para determinar o diâmetro do grafo $H_{l,p}$ temos que encontrar um vértice diametral, este vértice tem o valor máximo para a soma s_{l_1} em função da soma s_t e dos valores de l_1 e l_2 . Assim, para encontrar o vértice diametral, precisamos do valor máximo da soma s_{l_1} . Logo, o valor s_{l_1} depende da soma total s_t e do valor de l_1 . Para isso, apresentamos o Algoritmo 2.4, que encontra um vértice z do grafo $H_{l,p}$. Seja z um vetor inicial onde as coordenadas são zeros nos l_1 coordenadas e p nos l_2 coordenadas.

Algoritmo 2.4: Vértice z do grafo $H_{l,p}$ para o valor l_1

Entrada: l, p, l_1
Saída: z

```
1 início
2    $i = l_1; j = l_1 + 1$ 
3    $z = (0, 0, \dots, 0, p, p, \dots, p)$ 
4   enquanto  $(z[j] - 1) \bmod p \geq (z[i] + 1) \bmod p$  faça
5      $z[i] = z[i] + 1 \bmod p$ 
6      $z[j] = (z[j] - 1) \bmod p$ 
7      $i = i - 1; j = j + 1$ 
8     se  $i = 0$  então
9        $i = l_1$ 
10    se  $j = l + 1$  então
11       $j = l_1 + 1$ 
12  retorna  $z$ 
13 fim
```

Definição 2.27. Seja $z = (z_a, \dots, z_a, z_b, \dots, z_b) \in H_{l,p}$ um vértice retornado pelo Algoritmo 2.4 com valores de $l_a = (l_1 \cdot p) \bmod l$ e $l_b = l - l_a$, onde $z_b = z_a + 1$ ou $z_b \equiv_p 0$ e l_a é o número de coordenadas z_a e l_b é o número de coordenadas z_b .

A Tabela 2.2 mostra o resultado do Algoritmo 2.4 no grafo $H_{5,7}$ usando valores 2 e 3 para l_1 . Quando $l_1 = 2$ temos $s_{l_1} = 4 + 4 = 8$ e, quando $l_1 = 3$ temos $s_{l_1} = 2 + 3 + 3 = 8$.

Tabela 2.2: Vértice z do grafo $H_{5,7}$ com soma $s_{l_1} = 8$ para valores $l_1 = 2$ e $l_1 = 3$.

	$l_1 = 2$					$l_1 = 3$				
0	0	7	7	7		0	0	0	7	7
0	1	6	7	7		0	0	1	6	7
1	1	6	6	7		0	1	1	6	6
1	2	6	6	6		1	1	1	5	6
2	2	5	6	6		1	1	2	5	5
2	3	5	5	6		1	2	2	4	5
3	3	5	5	5		2	2	2	4	4
3	4	4	5	5		2	2	3	3	4
4	4	4	4	5		2	3	3	3	3

Por exemplo, no grafo $H_{5,7}$ usando o valor $l_1 = 2$, temos $l_a = 2 \cdot 7 \bmod 5 = 4$ e para $l_1 = 3$, temos $l_a = 3 \cdot 7 \bmod 5 = 1$.

Lema 2.28. Existe um vértice z no grafo $H_{l,p}$ para cada valor de l_1 que tem o valor máximo para a soma s_{l_1} .

Demonstração. Segue diretamente da construção do Algoritmo 2.4. □

Lema 2.29. *Seja z um vértice com valor máximo para a soma s_{l_1} para quaisquer valores de l_1 e l_2 do grafo $H_{l,p}$, então o valor de z_a é:*

$$z_a = \begin{cases} (l_2 \cdot (p-1) + (l_a - l_1))/l & \text{se } l_a \geq l_1, \\ ((p \cdot l_2) + l_a)/l - 1 & \text{se } l_a < l_1. \end{cases}$$

Demonstração. Seja $z = (z_a, \dots, z_a, z_b, \dots, z_b)$ um vértice com valor máximo para a soma s_{l_1} do grafo $H_{l,p}$, temos dois casos para analisar: a) quando $l_a \geq l_1$; b) quando $l_a < l_1$.

Caso a) temos:

$$s_t = l_1 \cdot z_a + (l_a - l_1) \cdot z_a + (l - l_a) \cdot z_b,$$

$$s_t = l_1 \cdot z_a + l_a \cdot z_a - l_1 \cdot z_a + l \cdot z_b - l_a \cdot z_b,$$

$$s_t = l_a \cdot z_a + l \cdot z_b - l_a \cdot z_b, \text{ fazendo } z_b = z_a + 1,$$

$$s_t = l_a \cdot z_a + l \cdot (z_a + 1) - l_a \cdot (z_a + 1),$$

$$s_t = l_a \cdot z_a + l \cdot z_a + l - l_a \cdot z_a - l_a,$$

$$s_t = l \cdot z_a + l - l_a,$$

$$l \cdot z_a = s_t - l + l_a, \text{ sabemos que } s_t = p \cdot (l - l_1),$$

$$l \cdot z_a = p \cdot (l - l_1) - l + l_a,$$

$$l \cdot z_a = p \cdot l - p \cdot l_1 - l + l_a, \text{ fazendo } l = l_1 + l_2,$$

$$l \cdot z_a = p \cdot (l_1 + l_2) - p \cdot l_1 - l_1 - l_2 + l_a,$$

$$l \cdot z_a = p \cdot l_1 + p \cdot l_2 - p \cdot l_1 - l_1 - l_2 + l_a,$$

$$l \cdot z_a = p \cdot l_2 - l_1 - l_2 + l_a,$$

$$l \cdot z_a = (l_2 \cdot (p-1) + (l_a - l_1)),$$

$$z_a = (l_2 \cdot (p-1) + (l_a - l_1))/l.$$

Caso b) temos:

$$s_t = l_a \cdot z_a + (l_1 - l_a) \cdot z_b + l_2 \cdot z_b,$$

$$s_t = l_a \cdot z_a + l_1 \cdot z_b - l_a \cdot z_b + l_2 \cdot z_b, \text{ fazendo } z_b = z_a + 1,$$

$$s_t = l_a \cdot z_a + l_1 \cdot (z_a + 1) - l_a \cdot (z_a + 1) + l_2 \cdot (z_a + 1),$$

$$s_t = l_a \cdot z_a + l_1 \cdot z_a + l_1 - l_a \cdot z_a - l_a + l_2 \cdot z_a + l_2,$$

$$s_t = l_1 \cdot z_a + l_1 - l_a + l_2 \cdot z_a + l_2, \text{ fazendo } l = l_1 + l_2,$$

$$\begin{aligned}
s_t &= l \cdot z_a + l - l_a, \\
s_t &= l \cdot (z_a + 1) - l_a, \\
l \cdot (z_a + 1) &= s_t + l_a, \text{ sabemos que } s_t = p \cdot (l - l_1), \\
l \cdot (z_a + 1) &= p \cdot (l - l_1) + l_a, \text{ fazendo } l_2 = l - l_1, \\
(z_a + 1) &= ((p \cdot l_2) + l_a)/l, \\
z_a &= (((p \cdot l_2) + l_a)/l) - 1.
\end{aligned}$$

□

Lema 2.30. *Seja z um vértice com valor máximo para a soma s_{l_1} para quaisquer valores de l_1 e l_2 do grafo $H_{l,p}$, então a distância $d(z)$ é:*

$$d(z) = \begin{cases} l_1 \cdot z_a & \text{se } l_a \geq l_1, \\ (l_1 \cdot (z_a + 1)) - l_a & \text{se } l_a < l_1. \end{cases}$$

Demonstração. Seja $z = (z_a, \dots, z_a, z_b, \dots, z_b)$ um vértice com valor máximo para a soma s_{l_1} do grafo $H_{l,p}$, temos dois casos para analisar: a) quando $l_a \geq l_1$; b) quando $l_a < l_1$.

Caso a) temos que $s_{l_1} = l_1 \cdot z_a$ e pelo Lema 2.26 $d = s_{l_1}$.

Caso b) temos:

$$\begin{aligned}
s_{l_1} &= l_a \cdot z_a + (l_1 - l_a) \cdot z_b, \\
s_{l_1} &= l_a \cdot z_a + l_1 \cdot z_b - l_a \cdot z_b, \text{ fazendo } z_b = z_a + 1, \\
s_{l_1} &= l_a \cdot z_a + l_1 \cdot (z_a + 1) - l_a \cdot (z_a + 1), \\
s_{l_1} &= l_a \cdot z_a + l_1 \cdot z_a + l_1 - l_a \cdot z_a - l_a, \\
s_{l_1} &= l_1 \cdot z_a + l_1 - l_a, \\
s_{l_1} &= (l_1 \cdot (z_a + 1)) - l_a.
\end{aligned}$$

□

Assim, podemos obter o diâmetro com o resultado seguinte.

Teorema 2.31. *Seja z um vértice com valor máximo para a soma s_{l_1} para quaisquer valores de l_1 e l_2 do grafo $H_{l,p}$. O diâmetro do grafo $H_{l,p}$ é $D = \max(d(z))$ para $1 \leq l_1 < l$.*

Demonstração. Pelo Lema 2.28, mostramos que o vértice z tem o valor máximo para a soma s_{l_1} do grafo $H_{l,p}$ para quaisquer valores de l_1 e l_2 . Pelo Lema 2.30, sabemos

a distância do vértice z para todos os valores de l_1 . Consideramos o valor de l_1 que encontra o valor máximo para $d(z)$, então é encontrado o diâmetro do grafo $H_{l,p}$. \square

Usamos o resultado do Teorema 2.31 para obtermos o Algoritmo 2.5. Este tem complexidade de tempo de $O(l)$. Para implementar o Algoritmo 2.5 usamos a linguagem PHP e o executamos em um computador com processador *i7*. Levamos apenas 24s para preencher a Tabela 2.1. Na versão anterior, levamos 24h para obter o mesmo resultado.

Algoritmo 2.5: Diâmetro do grafo $H_{l,p}$

Entrada: l integer, p integer
Saída: D integer

```

1 início
2    $D = 0; d = 0$ 
3   para  $i = 1$  until  $l - 1$  faça
4      $l_1 = i; l_2 = l - l_1$ 
5      $l_a = (l_1 \cdot p) \bmod l$ 
6     se  $l_a = 0$  então
7        $l_a = l$ 
8     se  $l_a \geq l_1$  então
9        $z_a = (l_2 \cdot (p - 1) + (l_a - l_1)) / l$ 
10       $d = l_1 \cdot z_a$ 
11     senão
12       $z_a = ((p \cdot l_2) + l_a) / l - 1$ 
13       $d = (l_1 \cdot (z_a + 1)) - l_a$ 
14     se  $d > D$  então
15       $D = d$ 
16   retorna  $D$ 
17 fim
```

2.4.2 Grafo $H'_{l,p}$

Nesta seção, mostramos como calcular o diâmetro dos grafos da família $H'_{l,p}$. Para o grafo $H'_{l,p}$, temos o seguinte resultado para a distância.

Lema 2.32. *A distância entre o vértice x do grafo $H'_{l,p}$ para o elemento identidade*

é $d(x) = l_1$, onde $l_1 = \sum_{i=1}^{l-1} \min(x_i, p - x_i)$.

Demonstração. Consideramos $x = \{x_1, x_2, \dots, x_l\}$ um vértice do grafo $H'_{l,p}$. Pela Definição 2.14 do conjunto gerador do grafo de Cayley $H'_{l,p}$, podemos somar ou subtrair 1 nas coordenadas entre 1 e $l - 1$. Assim, a menor distância entre a coordenada x_i e 0 é x_i ou $p - x_i$, sendo que no primeiro caso subtraímos x_i unidades e no segundo caso adicionamos $p - x_i$ unidades para chegar na identidade. Portanto,

$$d(x) = \sum_{i=1}^{l-1} \min(x_i, p - x_i). \quad \square$$

Pelo Teorema 2.33, determinamos o diâmetro do grafo $H'_{l,p}$, que é obtido como consequência do Lema 2.32.

Teorema 2.33. *O diâmetro do grafo $H'_{l,p}$ é $(\lfloor \frac{p}{2} \rfloor)(l - 1)$.*

Demonstração. Pelo Lema 2.32, temos que $d = \sum_{i=1}^{l-1} \min(x_i, p - x_i)$. Considerando o valor de $\lfloor \frac{p}{2} \rfloor$ para cada valor dos $l - 1$ coordenadas de um vértice diametral, obtemos que $D = (\lfloor \frac{p}{2} \rfloor)(l - 1)$. \square

2.5 Grafo $GC_n(b)$

O grafo $GC_n(b)$ foi definido por Lakshmivarahan, Jwo e Dhall em seu artigo sobre simetria em interconexão de redes baseado em grafo de Cayley do grupo das permutações [26]. Nesta seção, mostramos que o grafo $H'_{l,p}$ é isomorfo ao grafo $GC_{l-1}(p)$, para $p \leq 3$. Lakshmivarahan, Jwo e Dhall definiram o grafo de Cayley $GC_n(b) = (V, E)$ como o grafo hipercubo de base b e dimensão n [26], onde:

$$V = \{x_1, x_2, \dots, x_n \mid x_i \in \mathbb{Z}_b\} = (\mathbb{Z}_b)^n$$

$$E = \{(x, y) \mid x = x_1, x_2, \dots, x_n, y = y_1, y_2, \dots, y_n,$$

para algum i , $1 \leq i \leq n, x_i \neq y_i$ e $x_j = y_j$ para $j \neq i\}$.

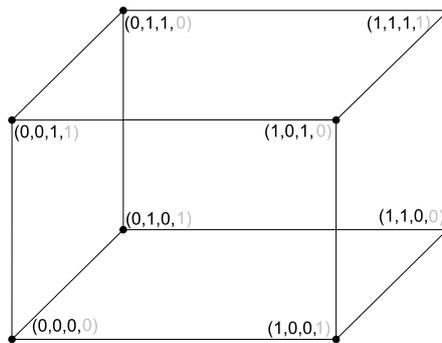


Figura 2.4: $GC_3(2) \cong H'_{4,2}$.

Seja $x, y \in V$. A distância de Hamming de base- b entre x e y denotada por $H_b(x, y)$ é o número de coordenadas em que x difere de y , isto é, a menor distância $d(x, y)$ é a distância de Hamming na base- b entre x e y . Assim, o diâmetro do grafo $GC_n(b)$ é n .

Pela definição do grafo $GC_n(b)$, podemos concluir que existe uma aresta entre dois vértices se, e somente se, uma das coordenadas dos vértices for diferente e o restante for igual. Com isso, concluímos que $GC_1(b)$ é isomorfo ao grafo completo com b vértices.

Pela definição do conjunto gerador $C'_{l,p}$ do grafo $H'_{l,p}$ podemos concluir que existe uma aresta entre dois vértices se, e somente se, dois coordenadas dos vértices forem diferentes, onde uma diferença se encontra nas $l - 1$ primeiras posições e a outra diferença se encontra na última posição. Portanto, temos que em alguns casos o grafo $H'_{l,p}$ será isomorfo a $GC_n(b)$, onde $l = n + 1$ e $p = b$. Quando consideramos os $l - 1$ valores para o vértice do grafo $H'_{l,p}$, temos a mesma definição de aresta do $GC_n(b)$.

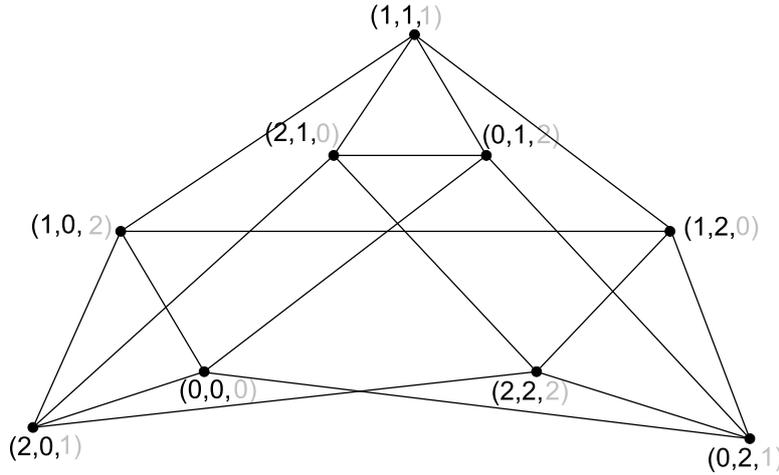


Figura 2.5: $GC_2(3) \cong H'_{3,3}$.

Lema 2.34. O grafo $H'_{l,p}$ é isomorfo ao grafo $GC_{l-1}(p)$, para $p = 2$ e $p = 3$.

Demonstração. Para o grafo $H'_{l,p}$, onde $p = b$, $b = 2$ e $l = n + 1$, temos que o grafo $H'_{l,2}$ é isomorfo ao grafo $GC_{l-1}(2)$, conforme exemplo da Figura 2.4 para $n = 3$ e $l = 4$. O grafo $GC_n(2)$ é o hipercubo n -dimensional binário [26].

Para o grafo $H'_{l,p}$, onde $p = b$, $b = 3$ e $l = n + 1$, temos que o grafo $H'_{l,3}$ é isomorfo ao $GC_{l-1}(3)$ conforme exemplo da Figura 2.5 para $n = 2$ e $l = 3$. \square

Vamos analisar se o grafo $H'_{l,p}$ é isomorfo a $GC_{l-1}(b)$ para valores $b > 3$. Sejam $x = (x_1, x_2, \dots, x_k, \dots, x_n)$ e $y = (y_1, y_2, \dots, y_k, \dots, y_n)$ dois vértices de $GC_n(b)$, onde $x_i = y_i$ para $1 \leq i \leq n$, $i \neq k$ e $x_k = 1$ e $y_k = 3$. Pela definição de $GC_{l-1}(b)$,

existe uma aresta entre x e y . Pela definição do grafo $H'_{l,p}$, não temos uma aresta entre x e y , pois temos uma aresta se, e somente se, $x_k = y_k + 1$ ou $x_k = y_k - 1$. Para $y_k = 3$ temos $x_k = 4$ ou $x_k = 2$, ou seja, valores diferentes de 1 e 3.

Vimos, neste capítulo, que os grafos das famílias $H_{l,p}$ e $H'_{l,p}$ são grafos de Cayley e mostramos como encontrar dois ciclos hamiltonianos. Um dos ciclos hamiltonianos das famílias $H_{l,p}$ e $H'_{l,p}$ é encontrado utilizando apenas $O(n)$ posições de memória. Os grafos $H_{l,p}$ e $H'_{l,p}$ são conexos, regulares, têm um diâmetro logarítmico e podem ser utilizados para criar interconexão de redes [23, 44].

Capítulo 3

Transposições Pré-fixadas Unitárias

Uma atividade frequente em Ciência da Computação consiste em comparar sequências e tem como principal objetivo o problema de distância, que é encontrar o número mínimo de operações que transformam uma sequência dada em outra. Mais recentemente, os eventos de reversões e transposições foram considerados, conhecidas como eventos de rearranjo, que consistem em comparar os genomas de duas espécies.

Um genoma arbitrário é formado por n genes e é representado por uma permutação de n elementos. O *Problema de Ordenação por Transposições* pede o número mínimo de transposições, eventos que trocam dois blocos adjacentes numa permutação, para transformar uma permutação na permutação identidade. Este problema é considerado desafiador em Biologia Computacional [3] e recentemente foi estabelecido como \mathcal{NP} -difícil [7]. Consequentemente, algumas variações deste problema têm sido estudadas, na esperança de que com isso surge alguma ideia sobre o problema de ordenação por transposições [6]. Uma variação do problema, chamado *Ordenação por Transposições Pré-Fixadas*, considera o problema de distância de rearranjo por transposições onde apenas as transposições que movem os primeiros elementos consecutivos do genoma são permitidos [12]. É um problema em aberto, com apenas alguns limites justos para a distância e para o diâmetro [9, 14, 25]. Aspectos teóricos envolvendo o grafo associado às permutações e suas distâncias de transposições pré-fixadas foram abordados por Reis [35]. Neste capítulo, consideramos as transposições pré-fixadas unitárias, que é o evento da transposição onde pelo menos um dos blocos trocados possui comprimento um.

Chamamos o grafo construído com as transposições pré-fixadas unitárias como *Grafo de Rearranjo por Transposições Pré-fixadas Unitárias*, denotado por $URG(n)$, cujos vértices são as permutações no grupo simétrico S_n e dois vértices são adjacentes se existe uma transposição pré-fixada unitária que aplicada a uma permutação produz a outra (Figura 3.1).

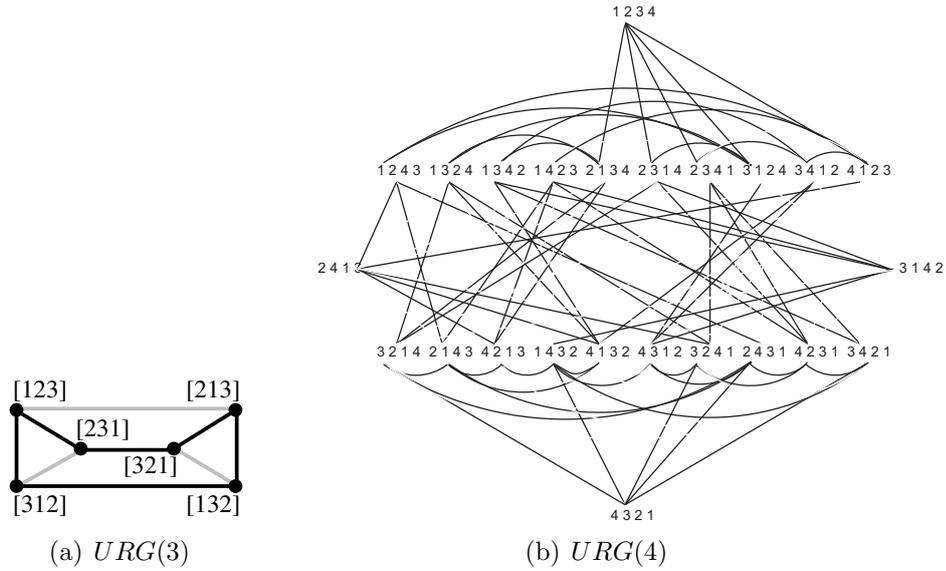


Figura 3.1: Os grafos $URG(3)$ e $URG(4)$.

Como já visto, os grafos de Cayley são vértice-transitivos. Estes grafos fornecem uma boa estrutura para o estudo de grupos abstratos, onde elementos são associados através de um conjunto gerador [26]. Em Biologia Molecular, os grafos de Cayley do grupo simétrico S_n são considerados, uma vez que permutações representam as sequências de genes nos cromossomos e genomas, e as operações nas permutações representam eventos evolutivos fornecidos pelos conjuntos geradores [23].

Determinar se um grafo possui um ciclo hamiltoniano é um problema \mathcal{NP} -Completo [21]. Uma conjectura atribuída a Lovász [28] afirma que todo grafo conexo e vértice-transitivo possui um caminho hamiltoniano.

Demonstramos que o grafo $URG(n)$ é um grafo de Cayley, portanto é vértice-transitivo. O estudo dos caminhos e ciclos hamiltonianos nestes grafos podem fornecer evidências adicionais para apoiar a conjectura Lovász, ou oferecer um contra-exemplo para refutar a conjectura. São conhecidos apenas quatro grafos de ordem maior que dois, vértice-transitivos e conexos que não possuem ciclo hamiltoniano, e todos estes grafos possuem caminho hamiltoniano. Eles são os grafos de Petersen, o grafo de Coxeter e os grafos obtidos a partir de cada um destes dois grafos, substituindo cada vértice por um triângulo. Como nenhum destes grafos são grafos de Cayley, um problema relacionado é determinar se todo grafo de Cayley associado a um grupo finito é hamiltoniano [23, 28].

3.1 Permutações

Nesta seção, apresentamos o grafo de Cayley pertencente ao grupo simétrico S_n . O conjunto de todas as permutações de n elementos, juntamente com a operação de

composição forma o grupo simétrico, denotado por S_n .

Para isso, temos algumas considerações: cada permutação é representada por uma função bijetiva; caso estejamos transformando uma permutação π em outra σ , então π e σ pertencem a S_n , ou seja, ambas possuem o mesmo número de elementos; Toda permutação $\pi \in S_n$ contém n elementos distintos; se π contém n elementos distintos então π é uma permutação do grupo simétrico S_n . Com isso, temos as seguintes definições:

Definição 3.1 (Permutação e Transposição [3, 6]). *Uma permutação $\pi_{[n]} = [\pi_1 \pi_2 \cdots \pi_n]$ é uma função bijetiva com domínio e imagem no conjunto $\{1, 2, \cdots, n\}$. Ou seja, $\pi(i) = \pi_i, \forall i = 1, 2, \cdots, n$ e $\pi(i) = \pi(j)$ se, e somente se, $i = j$.*

Uma transposição $t(i, j, k)$, onde $1 \leq i < j < k \leq n + 1$ é a permutação:

$$t(i, j, k) = [1 \ 2 \ \cdots \ i-1 \ j \ j+1 \ \cdots \ k-1 \ i \ \cdots \ j-1 \ k \ \cdots \ n].$$

Eventualmente, pode-se considerar a permutação com outros conjuntos de domínio e imagem, como por exemplo, o conjunto $\{0, 1, 2, \cdots, n-1\}$.

A transposição $t(i, j, k)$ “recorta” os elementos entre as posições j e $k-1$ (ambos incluídos) e “cola” imediatamente antes da i -ésima posição. Por exemplo, se $\pi = [\pi_1 \pi_2 \cdots \pi_{i-1} \ \boxed{\pi_i \cdots \pi_{j-1}} \ \boxed{\pi_j \cdots \pi_{k-1}} \ \pi_k \cdots \pi_n]$, então $\pi \cdot t(i, j, k) = [\pi_1 \pi_2 \cdots \pi_{i-1} \ \boxed{\pi_j \cdots \pi_{k-1}} \ \boxed{\pi_i \cdots \pi_{j-1}} \ \pi_k \cdots \pi_n]$.

Temos que uma permutação $\pi_{[n]}$ é um elemento do grupo simétrico S_n , cuja operação é o produto de permutações. Uma transposição $t(i, j, k)$ é um elemento do conjunto gerador de S_n , ou seja, toda permutação pode ser descrita por uma sequência de produtos de transposições. Para uma permutação $\pi_{[n]}$ chamamos *tamanho* da permutação $\pi_{[n]}$ pelo número n de elementos que existem em $\pi_{[n]}$, ou seja, $\pi_{[n]}$ possui tamanho n .

Quando não causar ambiguidade, escrevemos π para uma permutação pertencente à S_n omitindo o índice $[n]$. A transposição $t(i, j, k)$ aplicada em $\pi, \pi \cdot t(i, j, k)$, tem o efeito em π de tal maneira que o bloco entre as posições i e $j-1$ é trocado com o bloco entre as posições j e $k-1$, ou seja:

$$\pi \cdot t(i, j, k) = [\pi_1 \ \pi_2 \ \cdots \ \pi_{i-1} \ \boxed{\pi_j \ \cdots \ \pi_{k-1}} \ \boxed{\pi_i \ \cdots \ \pi_{j-1}} \ \pi_k \ \cdots \ \pi_n].$$

Exemplo 3.2. *Seja $\pi = [8 \ 5 \ \boxed{2 \ 10 \ 7 \ 4} \ \boxed{1 \ 9} \ 6 \ 3]$ uma permutação de S_{10} . Aplicando $t(3, 7, 9)$ em π obtemos: $\pi \cdot t(3, 7, 9) = [8 \ 5 \ \boxed{1 \ 9} \ \boxed{2 \ 10 \ 7 \ 4} \ 6 \ 3]$.*

Algumas permutações recebem nomes e símbolos especiais, como: *permutação identidade*, $\iota_{[n]} = [1 \ 2 \ \cdots \ n]$; *permutação reversa*, $\rho_{[n]} = [n \ n-1 \ \cdots \ 1]$. Outra definição importante é a de permutação inversa.

Definição 3.3 (Permutação inversa). *Dada uma permutação $\pi = [\pi_1 \pi_2 \cdots \pi_n]$, temos que $\pi^{-1} = [\pi_1^{-1} \pi_2^{-1} \cdots \pi_n^{-1}]$ é a permutação inversa de π , onde $\pi(\pi^{-1}(i)) = i$.*

Retornando ao Exemplo 3.2 temos que $\pi^{-1} = [7\ 3\ 10\ 6\ 2\ 9\ 5\ 1\ 8\ 4]$, isto devido a $\pi(\pi^{-1}(1)) = \pi(7) = 1$, $\pi(\pi^{-1}(2)) = \pi(3) = 2$, e assim por diante.

Um problema que surge neste contexto é determinar o valor mínimo de operações de transposição que transforme uma permutação em outra. Esta é portanto, a definição de distância de transposição, como segue.

Definição 3.4 (Distância de transposição). *A distância de transposição $d(\pi, \sigma)$ de uma permutação π a uma permutação σ , onde π e σ possuem o mesmo tamanho, é:*

- a) $d(\pi, \sigma) = 0$, se $\pi = \sigma$;
- b) $d(\pi, \sigma) = q$, sendo $q \geq 1$ o menor inteiro para o qual exista uma sequência de transposições t_1, t_2, \dots, t_q , tal que $\pi \cdot t_1 \cdot t_2 \cdots t_q = \sigma$, se $\pi \neq \sigma$.

É possível provar que dadas as permutações π, σ e γ , temos que $d(\pi, \sigma) = d(\gamma \cdot \pi, \gamma \cdot \sigma)$. Para isso considere $\pi \cdot t_1 \cdot t_2 \cdots t_q = \sigma$ e o produto $\gamma \cdot \pi \cdot t_1 \cdot t_2 \cdots t_q$. Pela associatividade do produto entre elementos de S_n , é imediato verificar que $\gamma(\pi \cdot t_1 \cdot t_2 \cdots t_q) = \gamma \cdot \sigma$.

Sabendo então que $d(\pi, \sigma) = d(\gamma \cdot \pi, \gamma \cdot \sigma)$, ao considerar $\gamma = \sigma^{-1}$ temos $d(\pi, \sigma) = d(\sigma^{-1} \cdot \pi, \iota)$. Assim, nosso problema de *Distância de transposição*, Definição 3.4, pode ser definido de maneira equivalente a determinar o tamanho q de uma sequência mínima de transposições t_1, t_2, \dots, t_q , tal que $\pi \cdot t_1 \cdot t_2 \cdots t_q = \iota$. Com isso, a distância de transposição de uma permutação π é computada em relação à ι , e escrevemos $d(\pi, \iota) = d(\pi)$. A partir da relação $d(\pi, \sigma) = d(\gamma \cdot \pi, \gamma \cdot \sigma)$, também verifica-se diretamente que $d(\pi) = d(\pi^{-1})$. Isto devido $d(\pi, \iota) = d(\pi^{-1} \cdot \pi, \pi^{-1}) = d(\iota, \pi^{-1})$.

Definição 3.5 (Diâmetro de transposição). *O diâmetro de transposição $TD(n)$ é o valor da maior distância de transposição dentre todas permutações com n elementos. Ou seja, $TD(n) = \max_{\pi \in S_n} \{d(\pi)\}$.*

Podemos representar uma permutação não apenas pela sua representação decimal, mas também pela sua representação binária. Com isso, temos o problema de Distância de Hamming, como segue.

Definição 3.6 (Distância de Hamming). *Sejam π_b e σ_b as representações binárias das permutações π e σ , respectivamente. A Distância de Hamming $d_H(\pi, \sigma)$ é o número de posições em π_b e σ_b com valores diferentes.*

Por exemplo, a permutação $\pi = [3, 1, 2, 0]$ tem a representação binária de $\pi_b = [11, 01, 10, 00]$ e a distância de Hamming $d_H(\pi, \iota)$ é 4.

3.2 Grafo de Rearranjo por Transposições Pré-fixadas Unitárias

Como visto na Seção 3.1 uma *permutação* $\pi = [\pi_1 \pi_2 \dots \pi_i \dots \pi_n]$ é uma função bijetiva de um conjunto finito sobre os seus elementos. Outra representação é a decomposição em *ciclos* disjuntos. Definimos o produto de duas permutações como uma composição para a direita. Portanto, o produto $\pi \cdot \sigma$ é a composição de duas funções, em que π é aplicado primeiro e depois σ .

O *grafo de rearranjo por transposições* [13, 17], denotado por $TRG(n)$, é um grafo de Cayley onde os vértices são as permutações em S_n e dois vértices π e σ são adjacentes se, e somente se, existe uma transposição $t(i, j, k)$, tal que $\sigma = \pi \cdot t(i, j, k)$ (conforme Figura 3.2(c) para $n = 4$).

Uma *transposição pré-fixada* [9, 12], denotada por $pt(j, k)$, é uma transposição $t(1, j, k)$, onde $1 < j < k \leq n+1$. O *grafo de rearranjo por transposições pré-fixadas*, denotado por $PRG(n)$, é um grafo de Cayley onde os vértices são as permutações em S_n e dois vértices π e σ são adjacentes se, e somente se, existe uma transposição pré-fixada $pt(j, k)$ tal que $\sigma = \pi \cdot pt(j, k)$ (conforme Figura 3.2(d) para $n = 4$).

Uma *transposição pré-fixada unitária*, denotada por $ut(2, k)$ ou $ut(k-1, k)$, é uma transposição pré-fixada $pt(j, k)$, com $j = 2$ ou $j = k-1$, e $2 < k \leq n+1$. O *grafo de rearranjo por transposições pré-fixadas unitárias*, denotado por $URG(n)$, é um grafo de Cayley onde os vértices são as permutações em S_n e dois vértices π e σ são adjacentes se, e somente se, existe um transposição pré-fixada unitária $ut(2, k)$ ou $ut(k-1, k)$ tal que $\sigma = \pi \cdot ut(2, k)$ ou $\sigma = \pi \cdot ut(k-1, k)$ (conforme Figura 3.2(a) para $n = 4$). O grafo $URG(n)$ está fortemente relacionado com outro grafo de transposições onde os $n!$ vértices são permutações em S_n .

O *grafo bubble-sort* [2], denotado por $BS(n)$, é um grafo de Cayley onde dois vértices π e σ são adjacentes se, e somente se, existe uma transposição $t(j-1, j, j+1)$, tal que $\sigma = \pi \cdot t(j-1, j, j+1)$ (conforme Figura 3.2(b) para $n = 4$). Observe que $BS(n)$ e $URG(n)$ são dois subgrafos geradores distintos do grafo $TRG(n)$.

O grafo $URG(n)$ é regular de grau $2n-3$ que é o número de transposições pré-fixadas unitárias distintas aplicáveis a uma permutação. O grafo $BS(n)$ tem grau $n-1$, o grafo de Cayley $PRG(n)$ tem grau $\frac{n^2-n}{2}$ e o grafo $TRG(n)$ tem grau $\frac{n^3-n}{6}$. Por exemplo, o grafo da Figura 3.1(a) é o grafo $URG(3)$ com conjunto gerador $\{213, 231, 312\}$.

Propriedade 3.7. *O grafo $URG(n)$ é um grafo de Cayley.*

Demonstração. Considere o subconjunto de P de S_n , onde $P = \{ut(j, k), j = 2 \text{ ou } j = k-1 \text{ e } 2 < k \leq n+1\}$. Observe que $\iota \notin P$, e que para cada permutação $ut(j, k)$ em P , temos a sua permutação inversa $ut(k-j+1, k)$ que também pertence

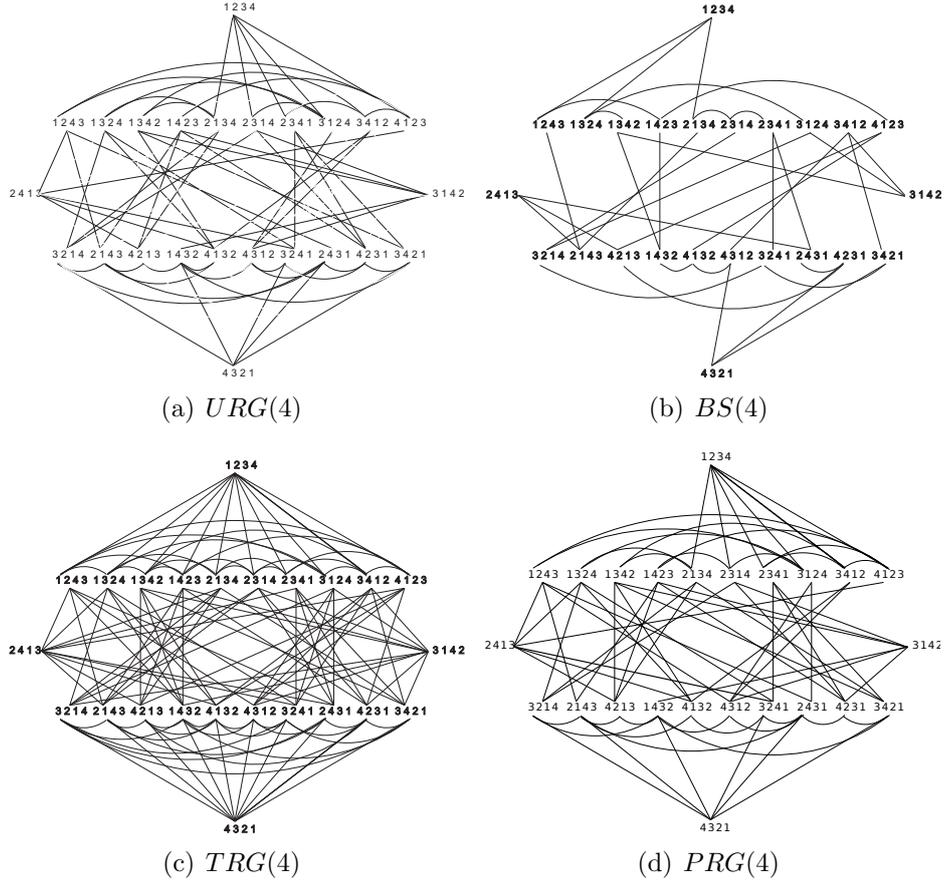


Figura 3.2: Os grafos de Cayley $BS(4)$, $URG(4)$, $PRG(4)$ e $TRG(4)$.

a P . Resta provar que P é um conjunto gerador de S_n . Cada permutação $\pi = [\pi_1 \ \pi_2 \ \dots \ \pi_n]$ em S_n pode ser escrita como produto de transposições pré-fixadas unitárias obtidas através do seguinte procedimento:

Para $i \leftarrow 1$ até $n - 1$ faça

seja l tal que $\pi_l = n - i$, faça $\pi \leftarrow \pi \cdot ut(l, l + 1)$.

Portanto, $URG(n)$ é um grafo de Cayley associado ao par (S_n, P) . \square

Temos portanto, como consequência da Propriedade 3.7, que o grafo $URG(n)$ é vértice-transitivo.

Propriedade 3.8. *O grafo $URG(n)$ não é aresta-transitivo.*

Demonstração. No grafo $URG(n)$, a aresta $e_1 = ([2 \ 3 \ 1 \ 4 \ 5 \ \dots \ n], [3 \ 2 \ 1 \ 4 \ 5 \ \dots \ n])$ não está em algum triângulo, mas a aresta $e_2 = ([1 \ 2 \ 3 \ 4 \ 5 \ \dots \ n], [3 \ 1 \ 2 \ 4 \ 5 \ \dots \ n])$ está em um triângulo. Segue que não há nenhum automorfismo A satisfazendo $A(e_1) = e_2$ e, portanto, $URG(n)$ não é aresta-transitivo. \square

Chamamos o subgrafo gerador pelo conjunto de vértices do grafo $URG(n)$ que representam as permutações na forma $[\pi_1 \ \pi_2 \ \dots \ \pi_{n-1} \ x]$, para $1 \leq x \leq n$, com $URG_x(n)$. Observe que todos os subgrafos de $URG_x(n)$ são disjuntos e a união de

todos os subgrafos contém todos os vértices do grafo $URG(n)$. Portanto, $URG(n)$ pode ser visto como o particionamento de n subgrafos $URG_x(n)$, para $1 \leq x \leq n$.

Definimos, usando a notação de [15], $\pi \oplus x$ com a permutação $[\pi'_1 \pi'_2 \dots \pi'_{n-1} x]$ onde

$$\pi'_i = \begin{cases} \pi_i, & \text{se } \pi_i \neq x \\ n, & \text{se } \pi_i = x. \end{cases}$$

Lema 3.9. Para $1 \leq x \leq n$, o grafo $URG_x(n)$ é isomorfo ao grafo $URG(n-1)$.

Demonstração. Sejam $\pi = [\pi_1 \dots \pi_{n-1}]$ e $\sigma = [\sigma_1 \dots \sigma_{n-1}]$ dois vértices distintos do grafo $URG(n-1)$. Suponha que existe uma transposição pré-fixada unitária $ut(j, k)$ com $\sigma = \pi \cdot ut(j, k) = [\pi_j \dots \pi_{k-1} \pi_1 \pi_{j-1} \pi_k \dots \pi_{n-1}]$, ou seja, $\{\pi, \sigma\}$ é uma aresta do grafo $URG(n-1)$. Para $1 \leq x \leq n$, observe que $\pi \oplus x$ e $\sigma \oplus x$ são vértices do grafo $URG_x(n)$ e que o conjunto de vértices do grafo $URG_x(n)$ corresponde a $(n-1)!$ permutações. Sejam $\sigma' = \sigma \oplus x$, $\pi' = \pi \oplus x$, e $\pi'' = \pi' \cdot ut(j, k)$ tal que l -ésimo elemento de σ , σ' e π'' são descritos nas equações.

$$\sigma_l = \begin{cases} \pi_{l+j-1}, & \text{se } 1 \leq l < 1+k-j \\ \pi_{l-k+j}, & \text{se } 1+k-j \leq l < k \\ \pi_l, & \text{se } l \geq k \end{cases} \quad \sigma'_l = \begin{cases} x, & \text{se } l = n \\ \sigma_l, & \text{se } \sigma_l \neq x \\ n, & \text{se } \sigma_l = x \end{cases}$$

$$\pi''_l = \begin{cases} \pi'_{l+j-1}, & \text{se } 1 \leq l < 1+k-j \\ \pi'_{l-k+j}, & \text{se } 1+k-j \leq l < k \\ \pi'_l, & \text{se } l \geq k. \end{cases}$$

Agora, comparamos σ' com π'' . A fim de mostrar que $\sigma' = \pi''$, vamos estabelecer que $\sigma' = (\pi \cdot ut(j, k)) \oplus x = (\pi \oplus x) \cdot ut(j, k) = \pi''$. Se $l = n$, então $\sigma'_n = \pi'_n = x$ e $k \leq n$, temos $\sigma'_n = \pi''_n$. Caso contrário, se $k \leq l < n$, então $\pi''_l = \pi'_l = \pi_l = \sigma_l = \sigma'_l$. Se $(1+k-j \leq l \leq k)$, então $\pi''_l = \pi'_{l-k+j}$ e existem dois casos: (a) $\sigma_l = x$ o que implica $\sigma'_l = n$ e, assim, $\sigma_l = \pi_{l-k+j}$ que por sua vez implica $\pi'_{l-k+j} = n$, e; (b) $\sigma_l \neq x$ o que implica $\pi_{l-k+j} \neq x$ e, assim, $\pi''_l = \pi'_{l-k+j} = \pi_{l-k+j} = \sigma_l = \sigma'_l$. O caso $(1 \leq l < 1+k-j)$ é análogo substituindo o índice $(l-k+j)$ por $(l+j-1)$. Como todos os elementos em σ' e π'' concordam, concluímos que $\sigma' = \pi''$. Portanto, a permutação $\pi \oplus x$ e $\sigma \oplus x$ são adjacentes em $URG_x(n)$ por $ut(j, k)$. \square

Mostramos, no Lema 3.9, que o grafo $URG(n)$ consiste em n cópias isomorfas do grafo $URG(n-1)$. Agora, mostramos como obter uma permutação com n elementos através de uma permutação com $(n-1)$ elementos. Seja $\pi = [\pi_1 \pi_2 \dots \pi_{n-1}]$ uma permutação de $n-1$ elementos e x é um número inteiro tal que $1 \leq x \leq n$.

Neste particionamento, podemos denominar $URG_x(n)$, $1 \leq x \leq n$, o grafo isomorfo ao $URG(n-1)$ com todas as permutações de n elementos contendo x na última posição. No grafo $URG(n)$ construído desta forma, cada vértice com último

elemento y possui as outras aresta incidente na partição $URG_x(n)$, $1 \leq x \leq n$, $x \neq y$. A Figura 3.3 exibe a construção de parte do grafo $URG(4)$ em 4 partições formadas por grafos isomorfos ao $URG(3)$ onde estão representadas apenas as arestas incidentes à permutação identidade ι .

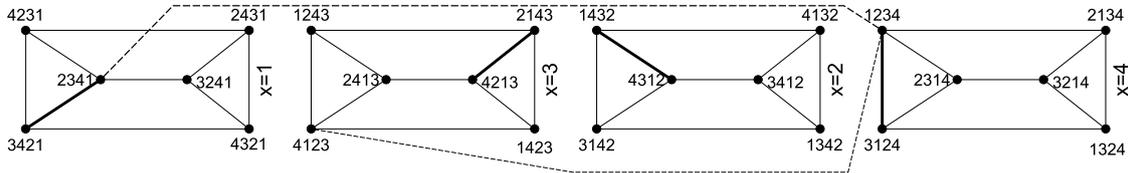


Figura 3.3: Parte do $URG(4)$ a partir do $URG(3)$ com todas as arestas incidentes ao vértice identidade ι .

3.3 Diâmetro

Nesta seção, abordamos o diâmetro do grafo $URG(n)$. Uma *maior subsequência crescente* é uma subsequência de uma sequência dada, em que os elementos na subsequência estão na ordem de classificação, menor para o maior, e a subsequência é máxima. Esta subsequência não é necessariamente contínua ou única.

Lema 3.10. *Cada transposição pré-fixada unitária aumenta em no máximo em 1 unidade o tamanho de l da maior subsequência crescente.*

Demonstração. Seja l o tamanho da maior subsequência crescente. Esta subsequência não é necessariamente contínua. Para as transposições pré-fixadas unitárias temos dois casos: a) $ut(k, k + 1)$ e b) $ut(2, k)$. No primeiro caso, movemos o elemento k para a primeira posição, portanto podemos adicionar em 1 ao valor de l , se o elemento k aumenta a subsequência. No segundo caso, movemos o elemento da primeira posição para a posição k , portanto podemos adicionar em 1 ao valor de l , se o primeiro elemento aumenta a subsequência. Em ambos os casos, temos que o tamanho l da maior subsequência crescente aumenta no máximo em 1 unidade. \square

Corolário 3.11. *Cada permutação $\pi \in S_n$ necessita no mínimo de $n - l$ transposições pré-fixadas unitárias para ser ordenada, onde l é o tamanho da maior subsequência crescente.*

Demonstração. Pelo Lema 3.10, cada transposição pré-fixada unitária aumenta o valor da maior subsequência crescente no máximo em 1 unidade. Segue que para ordenar uma permutação π , onde a maior subsequência crescente tem tamanho l , requer no mínimo $n - l$ transposições pré-fixadas unitárias. \square

Corolário 3.12. *A permutação reversa requer no mínimo de $n - 1$ transposições pré-fixadas unitárias para ser ordenada.*

Demonstração. Segue do Corolário 3.11 e do fato que a permutação reversa tem $l = 1$ como o tamanho da maior subsequência crescente. \square

Teorema 3.13. *O diâmetro do grafo $URG(n)$ é $D = n - 1$.*

Demonstração. Temos o limite superior pela Propriedade 3.7, esta propriedade ordena a permutação reversa usando $n-1$ transposições pré-fixadas unitárias. Também, podemos usar o resultado de Aigner e West [1] que diz que o diâmetro para o problema da distância do grafo de rearranjo que considera apenas a inserção do primeiro elemento, isto é, transposições da forma $t(1, 2, k)$ é $n - 1$. O limite inferior é obtido pelo Corolário 3.12, onde temos que a distância da permutação reversa é $d \geq n - 1$. \square

3.4 Ciclos úteis no grafo $URG(n)$

Um *ciclo útil* é um $2n$ -ciclo do grafo $URG(n)$ com $2n$ arestas. As primeiras n arestas contém precisamente uma aresta e_x de cada subgrafo $URG_x(n)$, $1 \leq x \leq n$. A Figura 3.4 mostra um ciclo útil realçado no grafo $URG(4)$. Cada partição $URG_x(4)$ é identificada por x e é isomorfa ao grafo $URG(3)$. As arestas entre vértices em partições diferentes são omitidas. Cada aresta e_x escolhida é chamada de *aresta representante* e esta aresta é associada a uma transposição pré-fixada unitária, chamada de *transposição pré-fixada unitária representante*, do tipo $ut(n - 1, n)$, denotada por $ut_r(n - 1)$ se n é par, caso contrário se n for ímpar, usamos uma transposição pré-fixada unitária do tipo $ut(n - 2, n - 1)$, denotada por $ut_r(n - 2)$. As outras n arestas são chamadas de *arestas ligantes*, elas estão associadas às *transposições pré-fixadas unitárias ligantes*, que são transposições pré-fixadas unitárias do tipo $ut(n, n + 1)$, denotado por $ut_l(n)$. As n arestas representantes e as n arestas ligantes alternam ao longo do ciclo útil. Mostramos no Lema 3.14 um ciclo útil para n par, e no Lema 3.15 um ciclo útil para n ímpar.

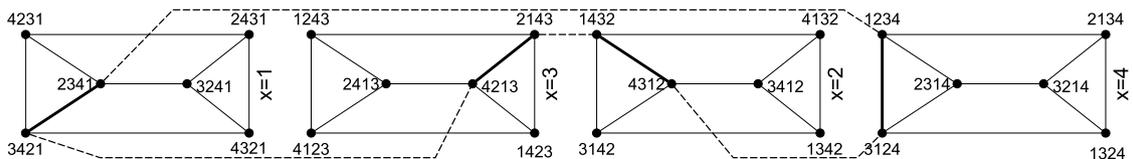


Figura 3.4: Um ciclo útil no grafo $URG(4)$. As arestas representantes associadas às $ut_r(3)$ são sólidas e as arestas ligantes associadas às $ut_l(4)$ são tracejadas.

Outra representação das permutações é conhecida por decomposição em ciclos disjuntos (veja em [14]). Nos Lemas 3.14 e 3.15 usamos as permutações de apenas um ciclo, denotadas por α , para definir a ordem de visitação nas partições $URG_x(n)$.

Lema 3.14. *Para $n \geq 4$ e par, existe um ciclo útil no grafo $URG(n)$ tal que $v_1 = \iota$ e $v_{2n} = \iota \cdot ut(2, n + 1) = [2\ 3 \ \dots \ n - 1\ n\ 1]$ aplicando $ut_r(n - 1)$ e $ut_l(n)$, alternadamente.*

Demonstração. Seja $\alpha = ut_r(n - 1) \cdot ut_l(n) = [n\ n - 1\ 1\ 2\ 3 \ \dots \ n - 2]$ uma permutação de apenas um ciclo $(n, n - 2, n - 4, \dots, 2, n - 1, n - 3, n - 5, \dots, 3, 1)$. Portanto, a ordem correspondente de cada grafo $URG_x(n)$, $1 \leq x \leq n$, no ciclo útil C é: $URG_n(n)$, $URG_{n-2}(n)$, $URG_{n-4}(n)$, \dots , $URG_2(n)$, $URG_{n-1}(n)$, $URG_{n-3}(n)$, $URG_{n-5}(n)$, \dots , $URG_1(n)$. Note que, ao percorrer o ciclo útil C até ι , a última aresta no ciclo útil é uma transposição do tipo $ut_l(n)$ e, assim, $v_{2n} \cdot ut_l(n) = \iota$. Uma vez que $ut(2, n + 1)$ é a inversa da transposição $ut_l(n)$. Portanto, temos que $v_{2n} = \iota \cdot ut(2, n + 1)$. \square

Lema 3.15. *Para $n \geq 5$ e ímpar, existe um ciclo útil no grafo $URG(n)$ tal que $v_1 = \iota$ e $v_{2n} = \iota \cdot ut(2, n + 1) = [2\ 3 \ \dots \ n - 1\ n\ 1]$ aplicando $ut_r(n - 2)$ e $ut_l(n)$, alternadamente.*

Demonstração. Seja $\alpha = ut_r(n - 2) \cdot ut_l(n) = [n\ n - 2\ 1\ 2\ 3 \ \dots \ n - 3\ n - 1]$ uma permutação de apenas um ciclo $(n, n - 1, n - 3, \dots, 2, n - 2, n - 4, n - 6, \dots, 3, 1)$. Portanto, a ordem correspondente de cada grafo $URG_x(n)$, $1 \leq x \leq n$, no ciclo útil C é: $URG_n(n)$, $URG_{n-1}(n)$, $URG_{n-3}(n)$, \dots , $URG_2(n)$, $URG_{n-2}(n)$, $URG_{n-4}(n)$, $URG_{n-6}(n)$, \dots , $URG_1(n)$. Note que, ao percorrer o ciclo útil C até ι , a última aresta no ciclo útil é uma transposição do tipo $ut_l(n)$ e, assim, $v_{2n} \cdot ut_l(n) = \iota$. Uma vez que $ut(2, n + 1)$ é a inversa da transposição $ut_l(n)$. Portanto, temos que $v_{2n} = \iota \cdot ut(2, n + 1)$. \square

A Figura 3.5 ilustra um ciclo útil no grafo $URG(5)$. As arestas representantes associadas às $ut_r(3)$ são sólidas e as arestas ligantes associadas às $ut_l(5)$ são tracejadas. Cada partição $URG_x(5)$ é identificada por um círculo cinza.

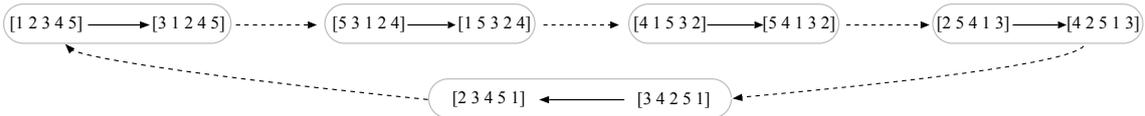


Figura 3.5: Um ciclo útil no grafo $URG(5)$.

3.5 Ciclo hamiltoniano no grafo $URG(n)$

Nesta seção, mostramos recursivamente como construir um ciclo hamiltoniano no grafo $URG(n)$, para $n \geq 4$, a partir de um ciclo útil no grafo $URG(n)$ e ciclos hamiltonianos no grafo $URG(n-1)$. Já que entre quaisquer dois elementos do ciclo útil, existem dois caminhos distintos, substituímos a aresta representante do ciclo útil do grafo $URG(n)$ por um caminho hamiltoniano adequado do grafo $URG(n-1)$. Discutimos dois casos complementares para n , pares e ímpares, que têm diferentes condições na construção do ciclo hamiltoniano. Em ambos os casos, usamos o ciclo hamiltoniano $URG(n-1)$ substituindo adequadamente as arestas representantes em um ciclo útil do grafo $URG(n)$, resultando em um ciclo hamiltoniano no grafo $URG(n)$. Pelo Lema 3.14, para $n \geq 4$ e par, a aresta representante é $ut_r(n-1)$, mas pelo Lema 3.15, para $n \geq 5$ e ímpar, a aresta representante é $ut_r(n-2)$. Portanto, para $n \geq 4$ e par, é necessário que o ciclo hamiltoniano no grafo $URG(n-1)$ tenha no mínimo uma aresta do tipo $ut(n-1, n)$, mas para $n \geq 5$ e ímpar, é necessário que o ciclo hamiltoniano no grafo $URG(n-1)$ tenha no mínimo duas arestas do tipo $ut(n-2, n-1)$. Observe que o grafo $URG(3)$ é hamiltoniano (Figura 3.1(a)) e que o ciclo hamiltoniano descrito tem a sequência das arestas $ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$ e $ut(2, 4)$. Tal que $URG(n)$ é um grafo vértice-transitivo, é irrelevante quais vértices estão ligados por essas arestas.

Teorema 3.16. *Para $n \geq 3$, $URG(n)$ tem um ciclo hamiltoniano C tal que:*

- (a) *Se n é par, então C tem no mínimo duas arestas $ut(2, n)$;*
- (b) *Se n é ímpar, então C tem no mínimo duas arestas $ut(2, n+1)$.*

Demonstração. Fazemos a prova por indução no valor de n . Para n ímpar, o caso base é $n = 3$. O grafo $URG(3)$ tem um ciclo hamiltoniano $C_o = (\iota = [1\ 2\ 3], [2\ 3\ 1], [3\ 2\ 1], [2\ 1\ 3], [1\ 3\ 2], [3\ 1\ 2])$. Este ciclo tem as arestas $([1\ 2\ 3], [2\ 3\ 1])$ e $([3\ 2\ 1], [2\ 1\ 3])$ que são $ut(2, n+1)$.

Para n par, o caso base é $n = 4$. O grafo $URG(4)$ tem o ciclo hamiltoniano: $C_e = (\iota = [1\ 2\ 3\ 4], [2\ 3\ 1\ 4], [3\ 2\ 1\ 4], [2\ 1\ 3\ 4], [1\ 3\ 2\ 4], [3\ 1\ 2\ 4], [4\ 3\ 1\ 2], [3\ 1\ 4\ 2], [1\ 3\ 4\ 2], [3\ 4\ 1\ 2], [4\ 1\ 3\ 2], [1\ 4\ 3\ 2], [2\ 1\ 4\ 3], [1\ 4\ 2\ 3], [4\ 1\ 2\ 3], [1\ 2\ 4\ 3], [2\ 4\ 1\ 3], [4\ 2\ 1\ 3], [3\ 4\ 2\ 1], [4\ 2\ 3\ 1], [2\ 4\ 3\ 1], [4\ 3\ 2\ 1], [3\ 2\ 4\ 1], [2\ 3\ 4\ 1] = \iota \cdot ut(2, 5))$. Este ciclo tem as arestas $([3\ 2\ 1\ 4], [2\ 1\ 3\ 4])$ e $([2\ 1\ 3\ 4], [1\ 3\ 2\ 4])$ que são $ut(2, n)$.

Suponha que o teorema seja válido para $n \geq 3$ e seja $m = n + 1$. Construimos um ciclo hamiltoniano no grafo $URG(m)$ a partir de um ciclo útil do grafo $URG(m)$ mantendo as m arestas ligantes como parte do ciclo hamiltoniano do grafo $URG(m)$ e substituindo as m arestas representantes pela mesma sequência das arestas de um caminho hamiltoniano na partição $URG_x(m)$, $1 \leq x \leq m$. Chamamos uma

sequência de transposições pré-fixadas unitárias aplicadas entre permutações consecutivas em um ciclo C por $S(C)$.

Para $m \geq 4$ e par, pelo Lema 3.14, existe um ciclo útil no grafo $URG(m)$, tal que $v_1 = \iota$ e $v_{2m} = \iota \cdot ut(2, m + 1)$, pela aplicação alternada da aresta $ut_r(m - 1)$ e da aresta $ut_l(m)$. Por hipótese de indução, existe um ciclo hamiltoniano C no grafo $URG(m - 1)$ que tem no mínimo duas arestas $ut(2, m)$.

Seja $S(C)$ uma sequência onde a última aresta é do tipo $ut(2, m)$ e e_x é uma aresta representante no grafo $URG_x(m)$, $1 \leq x \leq m$. Como $e_x = ut(m - 1, m)$ é uma aresta inversa de $ut(2, m)$, obtemos um ciclo hamiltoniano C' substituindo cada aresta e_x pela sequência de arestas $S(C)$ sem a última aresta $ut(2, m)$ no ciclo útil do grafo $URG(m)$. Como removemos somente a aresta $ut(2, m)$ da sequência $S(C)$ em cada uma das m substituições de e_x , C' tem no mínimo $m > 2$ arestas $ut(2, m)$.

Para $m \geq 5$ e ímpar, pelo Lema 3.15, existe um ciclo útil no grafo $URG(m)$, tal que $v_1 = \iota$ e $v_{2m} = \iota \cdot ut(2, m + 1)$, pela aplicação alternada da aresta $ut_r(m - 2)$ e da aresta $ut_b(m)$. Por hipótese de indução, existe um ciclo hamiltoniano C no grafo $URG(m - 1)$ que tem no mínimo duas arestas $ut(2, m)$.

Seja $S(C)$ uma sequência onde a última aresta é do tipo $ut(2, m - 1)$ e e_x é uma aresta representante no grafo $URG_x(m)$, $1 \leq x \leq m$. Como $e_x = ut(m - 2, m - 1)$ é a aresta inversa de $ut(2, m - 1)$, obtemos um ciclo hamiltoniano C' substituindo cada aresta e_x pela sequência de arestas $S(C)$ sem a última aresta $ut(2, m - 1)$ no ciclo útil do grafo $URG(m)$. Observe que, este ciclo útil tem m arestas ligantes $ut(m, m + 1)$. Percorrendo o ciclo C' na ordem reversa, obtemos a sequência $S(C')$ com m arestas $ut(2, m + 1)$. \square

Como um exemplo para o ciclo hamiltoniano construído pelo Teorema 3.16, vamos considerar a construção para $n = 4$. Veja a Figura 3.4, onde se destacam as arestas de um ciclo útil no grafo $URG(4)$. Cada aresta representativa e_x no grafo $URG(4)$, com $1 \leq x \leq 4$, é $ut(3, 4)$. Consideramos que o ciclo útil no grafo $URG(4)$ com a sequência de arestas $ut(3, 4)$, $ut(4, 5)$, $ut(3, 4)$, $ut(4, 5)$, $ut(3, 4)$, $ut(4, 5)$, $ut(3, 4)$ e $ut(4, 5)$ e o ciclo hamiltoniano C no grafo $URG(3)$ com $S(C) = ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$. Substituímos a sequência $S(C)$ sem a última aresta $ut(2, 4)$, e observe que a aresta inversa da aresta $ut(2, 4)$ é a aresta representativa $ut(3, 4)$, obtemos o ciclo hamiltoniano C' com a sequência $S(C') = ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$, $ut(4, 5)$, $ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$, $ut(4, 5)$, $ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$, $ut(4, 5)$, $ut(2, 4)$, $ut(2, 3)$, $ut(2, 4)$, $ut(2, 4)$, $ut(2, 3)$, $ut(4, 5)$, que é o ciclo hamiltoniano para o grafo $URG(4)$ mostrado na demonstração do Teorema 3.16.

Capítulo 4

Circuitos Reversíveis

Circuitos reversíveis possuem aplicações em Processamento Digital de Sinais, Comunicação, Computação Gráfica e Criptografia. Na teoria, os circuitos reversíveis podem ser implementados através de um sistema conservativo, ou seja, um sistema onde não há perda de calor. Na prática, há gasto de energia, mas esta é consideravelmente menor do que nos circuitos convencionais, nos quais apagar irreversivelmente um bit gasta no mínimo $kT \ln 2$, onde k é a constante de Boltzmann e T é a temperatura [27]. Com isso, computadores reversíveis podem ser economicamente mais interessantes do que um computador com capacidade de processamento equivalente pela economia de energia. Os circuitos reversíveis também são utilizados como base para os circuitos quânticos.

Em uma outra direção, a teoria dos grupos tem sido empregada como uma ferramenta para analisar as portas reversíveis e investigar os conjuntos geradores destas portas. Neste capítulo, estudamos a relação entre circuitos reversíveis e grafos de Cayley.

4.1 Circuitos reversíveis e quânticos

Um circuito lógico é composto de portas lógicas interconectadas. Uma porta lógica clássica é uma função $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ com n bits de entrada e m bits de saída. Nós definimos o *circuito combinacional* ou circuito irreversível como um circuito lógico acíclico, o que significa que cada instância da porta lógica é usado apenas uma vez.

Quando uma função f é bijetiva, f possui uma função inversa. Portanto, existe um circuito que, para cada valor de saída y de f , produz o valor de x de modo que $f(x) = y$. Neste caso, dizemos que o circuito é reversível. Uma n -porta reversível realiza uma função bijetiva sobre a permutação $\{0, 1, \dots, 2^n - 1\}$. Para qualquer porta reversível g , a porta g^{-1} realiza a transformação inversa. Cada permutação é uma sequência de $n2^n$ bits.

Uma *porta Toffoli generalizada* ou porta G-Toffoli $C^n NOT(x_1, x_2, \dots, x_n)$ mantém as primeiras $n - 1$ linhas, chamadas de linhas de controles, inalteradas. Esta porta inverte a n -ésima linha, chamada de linha alvo, se e somente se, cada linha de controle tem valor igual a 1. Por exemplo, a Figura 4.1 mostra uma porta $C^4 NOT(a, b, c, d)$, a linha do topo denota o bit menos significativo. Para $n = 0, 1, 2$ as portas são nomeadas por NOT (ou N), $CNOT$ (ou C), e Toffoli (ou T), respectivamente (veja Figura 4.2). Estas três portas compõem a biblioteca CNT [43], que é um conjunto universal de portas para a computação clássica reversível.

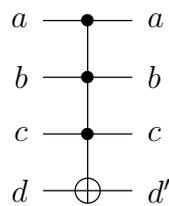


Figura 4.1: Porta G-Toffoli representando $C^4 NOT(a, b, c, d)$. A linha do topo denota o bit menos significativo.

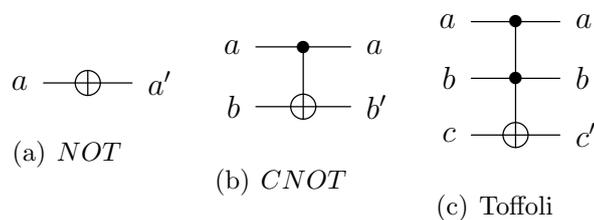


Figura 4.2: Circuito representando a biblioteca de portas CNT .

Observe que uma n -porta reversível aplicada em uma posição específica obtém uma permutação em S_{2^n} . Por exemplo, com a notação decimal, aplicando a porta N sobre uma linha da permutação temos $[1\ 0]$. Se a porta N é aplicada sobre o bit mais significativo em um circuito de 2 linhas temos a permutação $[2\ 3\ 0\ 1]$. Se a porta $CNOT$ é aplicada sobre um circuito de 2 linhas obtemos a permutação $[0\ 1\ 3\ 2]$ ou a permutação $[0\ 3\ 2\ 1]$, dependendo da posição do bit de controle.

A concatenação de portas de um circuito é equivalente a realização da composição de permutações associados a cada porta concatenada na mesma ordem.

Definição 4.1. *Seja L uma biblioteca de portas reversíveis. Um L -circuito é um circuito composto apenas de portas da biblioteca L . A permutação $\pi \in S_{2^n}$ é L -construtível se puder ser obtido por um L -circuito.*

Teorema 4.2 (Shende et al. [42]). *Toda permutação é CNT -construtível, com no máximo uma linha de armazenamento temporário.*

Definição 4.3. L_I é a biblioteca de portas reversíveis formada apenas por portas Toffoli generalizadas.

Uma porta Toffoli de controles mistos ou porta CM-Toffoli $C^n NOT(x_1, x_2, \dots, x_n)$ mantém as primeiras $n - 1$ linhas, chamadas de linhas de controles, inalteradas. Esta porta inverte a n -ésima linha, chamada de linha alvo, se e somente se cada linha de controle positivo (ou negativo) tem valor igual a 1 (ou 0). Indica-se a linha que está com controle negativo colocando ' após o controle. Veja Figura 4.3 para um exemplo de uma porta Toffoli de controles mistos 4×4 com um padrão negativo-positivo-negativo de linhas de controles e de alvo na última linha, que pode ser denotada por $C^4(a', b, c', d)$.

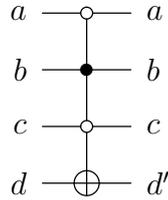


Figura 4.3: Porta CM-Toffoli representada por $C^4 NOT(a', b, c', d)$. A linha do topo denota o bit menos significativo.

Teorema 4.4 (Toffoli [43]). *Qualquer função reversível finita de ordem n é obtida pela composição de várias portas Toffoli de controles mistos, utilizando exatamente $n - 1$ controles.*

Definição 4.5. L_M é a biblioteca de portas reversíveis formada apenas por portas Toffoli de controles mistos, utilizando exatamente $n - 1$ controles.

Na síntese de circuito quântico, um pequeno conjunto de portas primitivas são usadas para a construção de blocos elementares com um custo unitário assumido [4, 31, 45]. Um conjunto padrão de portas universais é composta pelas portas: Hadamard, phase, CNOT and $\pi/8$ [34]. No contexto de nosso trabalho, também é razoável incluir neste conjunto as portas: NOT, a porta controlada- V , e a porta controlada- V^\dagger . Definimos V como a raiz quadrada da porta NOT, ou seja, um operador unitário, tal que V^2 é igual ao operador NOT. Cada porta Toffoli, porta G -Toffoli, ou porta CM -Toffoli pode ser decomposta em uma sequência de portas quânticas do conjunto acima referido, seguindo o padrão das Figuras 4.4 e 4.5. Um padrão de decomposição análogo é possível para portas quânticas com mais de dois controles [34].

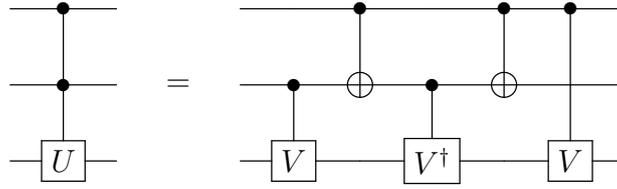


Figura 4.4: Decomposição de uma porta quântica com dois controles em uma sequência de portas quânticas com um único controle.

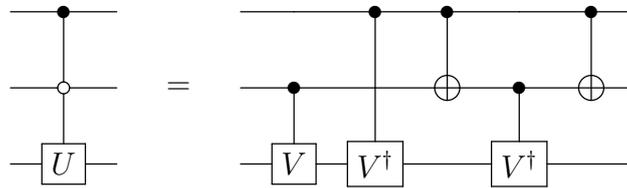


Figura 4.5: Decomposição de uma porta quântica com dois controles, sendo um controle negativo, em uma sequência de portas quânticas com um único controle.

O número de portas tem sido utilizado na literatura como medida para avaliar as abordagens de sínteses. Para um circuito arbitrário C que consiste numa sequência p_1, p_2, \dots, p_k de k portas quânticas, a métrica *número de portas* é definido como $np(C) \equiv k$. Nós também referimos à noção de *custo quântico* para medir o custo da implementação dos circuitos quânticos. Mais precisamente, o custo quântico é definido como o número de operações elementares quânticas necessárias para realizar uma porta. Para uma porta quântica arbitrária p que pode ser decomposta em l portas quânticas elementares, a métrica do seu *custo quântico* é definida como $cq(p) \equiv l$. O custo quântico para um circuito C é definido como $cq(C) = \sum_{p_i \in C} cq(p_i)$.

A Tabela 4.1 mostra o custo quântico para todas as portas reversíveis usadas neste trabalho, com m denotando a quantidade de controles negativos nas portas CM -Toffoli.

Tabela 4.1: Custo quântico de portas reversíveis.

tipo de porta	tamanho	lixo	custo quântico
NOT	1	0	1 [34]
$CNOT$	2	0	1 [34]
Toffoli	3	0	5 [4]
Toffoli com um controle negativo	3	0	5
Toffoli com um controle negativo	3	0	7
G -Toffoli			
$C^n NOT(x_1, x_2, \dots, x_n)$	n	0	$2^n - 3$ [4]
$C^n NOT(x_1, x_2, \dots, x_n)$	n	1	$24n - 88$ [4, 31]
$C^n NOT(x_1, x_2, \dots, x_n)$	n	$n - 3$	$10n - 25$ [24, 30]
CM -Toffoli			
$C^n NOT(x_1, x_2, \dots, x_n)$	n	0	$2^n - 3 + 2m$
$C^n NOT(x_1, x_2, \dots, x_n)$	n	1	$24n - 86$
$C^n NOT(x_1, x_2, \dots, x_n)$	n	$n - 3$	$10n - 23$

A Figura 4.6 mostra um exemplo da decomposição de uma porta CM -Toffoli, com $n = 6$, em portas Toffoli elementares. Esta decomposição utiliza uma porta com 5 linhas controles, três linhas de lixo e uma linha para o alvo. Este método de síntese foi baseado nos trabalhos de Kowada [24] e Maslov e Saeedi [30]. Neste caso, o custo quântico é o número de portas multiplicado por 5.

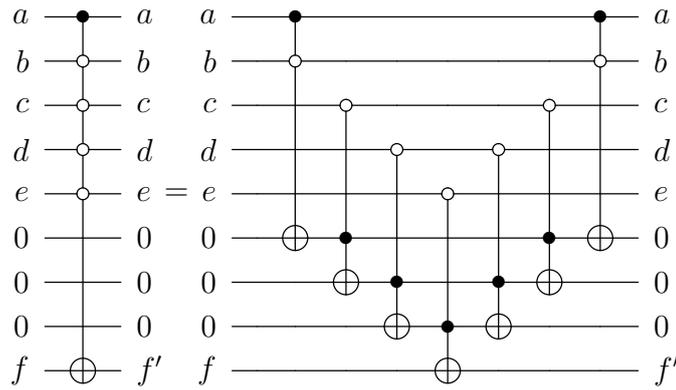


Figura 4.6: Implementação da porta CM -Toffoli para $n = 6$ e três linhas de lixo baseada nos trabalhos de Kowada [24] e Maslov e Saeedi [30].

A Figura 4.7 mostra um exemplo da decomposição de uma porta CM -Toffoli, com $n = 6$, em portas Toffoli elementares. Esta decomposição utiliza uma porta com 5 linhas controles, três linhas de lixo e uma linha para o alvo. Este método de

síntese foi baseado nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31]. Neste caso, observe que não é necessário inicializar as linhas de lixo com o valor zero.

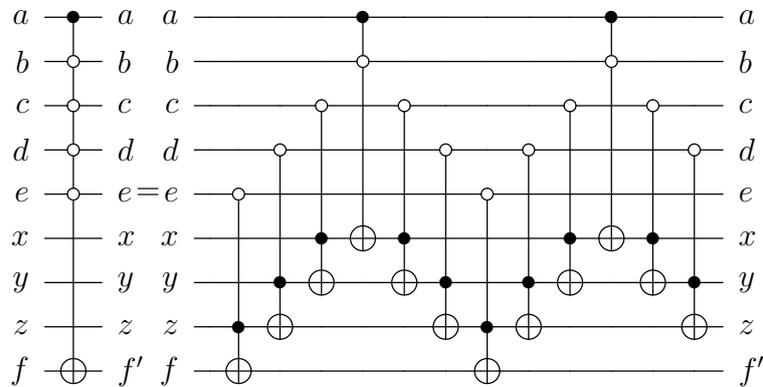


Figura 4.7: Implementação da porta CM-Toffoli para $n = 6$ e três linhas de lixo baseada nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31].

A Figura 4.8 mostra um exemplo da decomposição de uma porta CM-Toffoli de tamanho $n = 8$ e com uma linha de lixo em quatro portas CM-Toffoli com três linhas de lixo. Na Figura 4.9, é mostrada a decomposição das quatro portas CM-Toffoli com três linhas de lixo em portas Toffoli elementares de acordo com a decomposição da Figura 4.7, através do método de síntese baseado nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31].

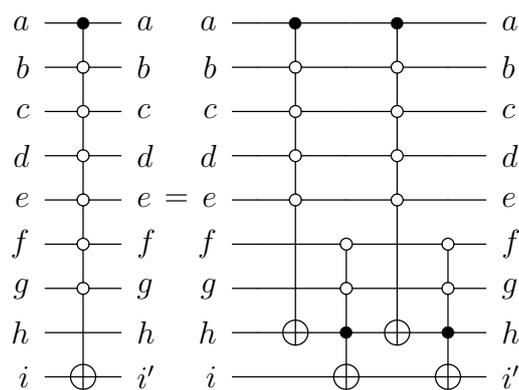


Figura 4.8: Implementação da porta CM-Toffoli para $n = 8$ com uma linha de lixo baseada nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31].

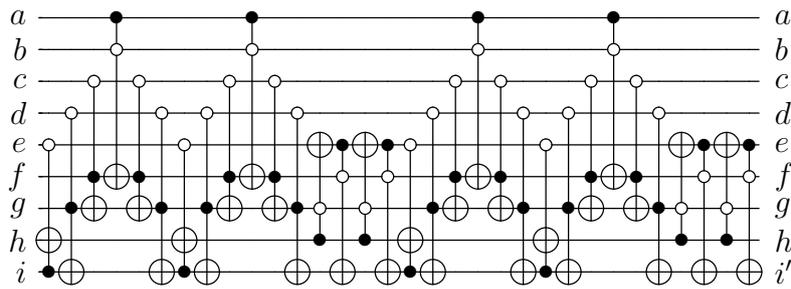


Figura 4.9: Decomposição da porta CM -Toffoli dada na Figura 4.8 em portas elementares de 3-bit.

A Figura 4.10 mostra um exemplo da decomposição de uma porta G-Toffoli, com $n = 9$, em portas Toffoli elementares. Esta decomposição utiliza uma porta com 8 linhas controles, sem linhas de lixo e uma linha para o alvo. Este método de síntese foi baseado nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31].

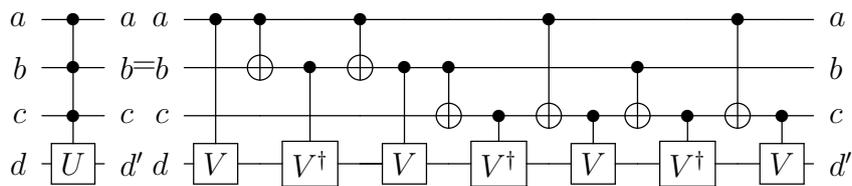


Figura 4.10: Implementação da porta G-Toffoli para $n = 4$ e sem linhas de lixo baseada nos trabalhos de Barenco *et al.* [4] e Maslov *et al.* [31].

4.2 Síntese de circuito usando portas G-Toffoli

Demonstramos a seguir um método de síntese de circuitos reversíveis utilizando as portas Toffoli generalizadas, proposto por Miller, Maslov e Dueck, que chamamos de *método MMD*. Eles propuseram este método em 2003 [32, 33] e reproduzimos o método no *Algoritmo 4.1*. O algoritmo considera uma função reversível especificando um mapeamento sobre os elementos da permutação $\{0, 1, \dots, 2^n - 1\}$, ou seja, sobre a tabela verdade. O algoritmo escreve uma função $f(i)$, onde i é um número inteiro na faixa de $0 \leq i \leq 2^n - 1$, o que significa que o argumento da função i é um vetor com a expansão binária do número inteiro i . O resultado da aplicação da função no argumento do número inteiro i , $f(i)$, também é tratado como um número inteiro. O algoritmo MMD trabalha atribuindo portas G-Toffoli na saída, colocando estas

portas em cascata. As portas G-Toffoli são escolhidas de modo que a permutação de saída é progressivamente transformada para coincidir com a permutação de entrada. Quando a permutação identidade for encontrada, o algoritmo lê a cascata de portas G-Toffoli na ordem inversa para transformar a permutação de entrada na permutação de saída.

Algoritmo 4.1: MMD [32, 33]

1 **início**

- 2 **Passo 0:** Se $f(0) \neq 0$, inverte as saídas correspondentes aos 1-bits em $f(0)$. Cada inversão requer uma porta NOT. A função de transformação, escrito como f^+ , tem $f^+(0) = 0$.
- 3 **Passo i:** Considere cada i sendo um valor $0 \leq i \leq 2^n - 1$ e f^+ denota a especificação reversível atual. Se $f^+(i) = i$, nenhuma transformação e, conseqüentemente, nenhuma porta G-Toffoli é necessária para este i . Caso contrário, portas são necessárias para transformar a especificação em uma nova especificação f^{++} com $f^{++}(i) = i$. As portas necessárias devem mapear $f^+(i) \rightarrow i$.
- 4 - Seja p a seqüência de bits com 1s em todas as posições em que a expansão binária de i é 1, enquanto a expansão de $f^+(i)$ é 0. Estes são os 1 bits que devem ser adicionados na transformação $f^+(i) \rightarrow i$. Por outro lado, seja q a seqüência de bits com 1s em todas as posições em que a expansão binária de i é 0, enquanto a expansão de $f^+(i)$ é 1. q identifica o 1 bits que serão removidos na transformação.
- 5 - Para cada $p_j = 1$, aplique a porta G-Toffoli com linhas de controle correspondentes a todas as saídas em posições onde a expansão de i é 1 e cujo alvo é a linha de saída na posição j . Isto irá aumentar a ordem lexicográfica $f^+(i)$. Então, para cada $q_k = 1$, aplique a porta G-Toffoli cujo alvo é a linha de saída na posição k , e com linhas de controle correspondentes a todas as saídas em posições, exceto k , onde a expansão de $f^+(i)$ é 1. Esta segunda operação diminui a ordem lexicográfica, mas não abaixo de i .

6 **fim**

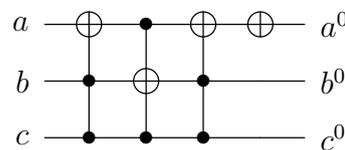


Figura 4.11: Circuito reversível que transforma permutação ι em $\pi = [1\ 0\ 3\ 2\ 5\ 7\ 4\ 6]$, de acordo com o exemplo da Tabela 4.2.

A Tabela 4.2 ilustra a aplicação do algoritmo MMD. Note-se que as portas estão identificados na ordem da saída para a entrada. O circuito correspondente é mostrado na Figura 4.11.

Tabela 4.2: Exemplo da aplicação do algoritmo MMD.

	(i)	(ii)	(iii)	(iv)	(v)
cba	$c^0b^0a^0$	$c^1b^1a^1$	$c^2b^2a^2$	$c^3b^3a^3$	$c^4b^4a^4$
000	001	000	000	000	000
001	000	001	001	001	001
010	011	010	010	010	010
011	010	011	011	011	011
100	101	100	100	100	100
101	111	110	111	101	101
110	100	101	101	111	110
111	110	111	110	110	111

Tabela 4.3: Número de funções reversíveis 3×3 para o Algoritmo 4.1 usando portas G -Toffoli e comparando com os melhores resultados a partir de Shende *et al.* [42].

Tamanho	Número de permutações	
	Algoritmo 4.1	resultados ótimos usando a biblioteca <i>CNT</i>
17	1	
16	14	
15	92	
14	380	
13	1113	
12	2468	
11	4311	
10	6083	
9	7044	
8	6754	577
7	5379	10253
6	3549	17049
5	1922	8921
4	839	2780
3	286	625
2	72	102
1	12	12
0	1	1
média de portas	8.67	5.63

A Tabela 4.3 mostra os resultados da aplicação da versão do Algoritmo 4.1 sobre todas as $8! = 40320$ permutações quando $n = 3$. Na primeira coluna temos o número de portas G-Toffoli utilizadas, na segunda coluna temos o total de permutações encontradas aplicando o Algoritmo 4.1 e a terceira coluna temos o resultado ótimo usando as portas *CNT*. Para cada cenário, mostra o número médio de portas G-Toffoli necessárias. Usando o algoritmo MMD, é possível encontrar uma permutação para qualquer valor de n que requer no máximo $(n - 1)2^n + 1$ portas G-Toffoli. O algoritmo encontra a permutação $[7\ 1\ 4\ 3\ 0\ 2\ 6\ 5]$ para $n = 3$ e a permutação $[15\ 1\ 12\ 3\ 5\ 6\ 8\ 7\ 0\ 10\ 13\ 9\ 2\ 4\ 14\ 11]$ para $n = 4$.

4.3 Síntese de circuitos usando portas CM-Toffoli

Nesta seção, apresentamos a seguir a síntese circuito reversível usando as portas Toffoli de controles mistos. A aplicação da porta Toffoli com controles mistos sobre a permutação π gera a permutação π' , onde temos a mudança em dois bits, ou seja, a sua distância de Hamming é $d_H(\pi, \pi') = 2$. Em relação a permutação identidade ι , temos três casos para a distância de Hamming: i) $d_H(\pi_b, \iota_b) = d_H(\pi'_b, \iota_b) - 2$, quando a porta coloca dois bits em suas posições corretas; ii) $d_H(\pi_b, \iota_b) = d_H(\pi'_b, \iota_b)$, quando a porta coloca um bit em sua posição correta e coloca o outro bit em uma posição errada; iii) $d_H(\pi_b, \iota_b) = d_H(\pi'_b, \iota_b) + 2$, quando a porta coloca os dois bits em posições erradas.

Para as portas CM-Toffoli apresentamos dois métodos de síntese de circuito que utilizam as propriedades i) e ii) da distância de Hamming. O primeiro método apresentado na Seção 4.3.1, chamado de *método Hipercubo*, garante a propriedade de sempre colocar um bit em seu lugar a cada aplicação de uma porta CM-Toffoli. O segundo método apresentado na Seção 4.3.2, chamado de *método de controles mistos*, garante que a cada iteração do algoritmo o k -ésimo bit mais significativo é colocado em seu respectivo lugar.

4.3.1 Método Hipercubo

O método Hipercubo para síntese de circuito reversível utiliza a biblioteca de porta L_M . Este método, apresentado no Algoritmo 4.2, usa aplicações consecutivas de portas CM-Toffoli para organizar os bits. O método proposto utiliza a representação binária dos elementos de permutação—cada elemento é composto de n bits—e realiza no máximo n trocas. Estas trocas utilizam as portas CM-Toffoli para colocar cada bit do elemento da permutação na sua posição correta. Veja o exemplo na Tabela 4.4. O circuito reversível correspondente é mostrado na Figura 4.12. Os elementos que serão trocados são apresentados em negrito. Os elementos ordenados

são apresentados sublinhados.

Tabela 4.4: Evolução da permutação π sendo transformada em ι pelo método Hipercubo.

	elementos da permutação							
porta aplicada	7	4	1	0	3	2	6	5
	111	100	001	000	011	010	110	101
passo $i = 7$ $C^3NOT(a, c, b)$	111	100	001	000	011	010	110	101
	101	100	001	000	011	010	110	<u>111</u>
passo $i = 6$	101	100	001	000	011	010	<u>110</u>	<u>111</u>
passo $i = 5$ $C^3NOT(b, c', a)$	101	100	001	000	011	010	<u>110</u>	<u>111</u>
$C^3NOT(a, c', b)$	101	100	001	000	010	011	<u>110</u>	<u>111</u>
$C^3NOT(a, b', c)$	101	100	011	000	010	001	<u>110</u>	<u>111</u>
	001	100	011	000	010	<u>101</u>	110	111
passo $i = 4$ $C^3NOT(a', c', b)$	001	100	011	000	010	<u>101</u>	<u>110</u>	<u>111</u>
$C^3NOT(a', b', c)$	001	100	011	010	000	<u>101</u>	110	111
	001	000	011	010	<u>100</u>	<u>101</u>	110	111
passo $i = 3$ $C^3NOT(b, c', a)$	001	000	011	010	<u>100</u>	<u>101</u>	110	111
	001	000	010	<u>011</u>	<u>100</u>	<u>101</u>	110	111
passo $i = 2$	001	000	<u>010</u>	<u>011</u>	<u>100</u>	<u>101</u>	110	111
passo $i = 1$ $C^3NOT(b', c', a)$	001	000	<u>010</u>	<u>011</u>	<u>100</u>	<u>101</u>	110	111
	<u>000</u>	<u>001</u>	<u>010</u>	<u>011</u>	<u>100</u>	<u>101</u>	110	111
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>

Por exemplo, a Tabela 4.4 mostra a execução do método Hipercubo na transformação da permutação $\pi = [7\ 4\ 1\ 0\ 3\ 2\ 6\ 5]$ na permutação identidade $\iota = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$.

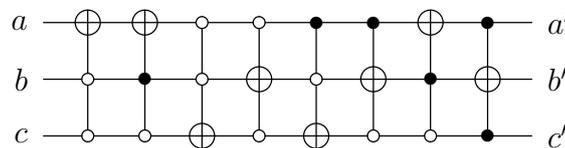


Figura 4.12: Circuito reversível que transforma permutação $\pi = [7\ 4\ 1\ 0\ 3\ 2\ 6\ 5]$, de acordo com o exemplo da Tabela 4.4.

O Algoritmo 4.2 faz a síntese de circuito reversível usando as portas CM-Toffoli. A permutação é lida na ordem da direita para a esquerda, chamamos de *ordem direita*. Pode-se alterar a linha 4 do Algoritmo 4.2 para ler a permutação na ordem

da esquerda para a direita. Neste caso, chamamos de *ordem esquerda*. Chamamos de *unidirecional* se o Algoritmo 4.2 roda apenas na ordem direita ou ordem esquerda. Chamamos de *bidirecional* se o Algoritmo 4.2 roda simultaneamente em ambas as ordens.

Algoritmo 4.2: Hiper cubo

```

entrada:  $\pi_b$  vetor
saída : circuito pilha
1 início
2    $N =$  tamanho do vetor  $\pi_b$ 
3    $n = \log_2 N$ 
4   para  $i = N - 1$  até 1 faça
5     se  $\pi_b[i] \neq \iota_b[i]$  então
6       para  $j = n - 1$  até 0 faça
7         se  $\pi_b[i][j] \neq \iota_b[i][j]$  então
8           % Adicione ao circuito a porta CM-Toffoli com
           alvo na posição  $j$  e controles nas posições  $\pi_b - j$ 
           coloque  $C^n NOT(\pi_b[0], \dots, \pi_b[n - 1], \pi_b[j])$  no circuito
           troque  $\pi_b[i][j]$ 
9 fim

```

A seguir, apresentamos uma propriedade existente na mudança dos bits da permutação no método Hiper cubo. Esta propriedade é utilizada na prova de correção do Algoritmo 4.2.

Propriedade 4.6. Quando $i < k$ no laço externo do Algoritmo 4.2, seja π uma permutação qualquer e seja $\pi[k]$ um elemento da permutação π , onde $i < k \leq 2^n - 1$. Então temos que $\pi[k] = k$.

Teorema 4.7. O Algoritmo 4.2 retorna um circuito reversível usando portas CM-Toffoli.

Demonstração. Vamos provar por indução a correção do Algoritmo 4.2, mostrando que, na aplicação de cada porta CM-Toffoli, pelo menos um bit troca ou fica na sua posição correta. Hipótese de indução: no passo $i = k$ e $j = l$, temos $\pi[k'] = k'$ e $\pi_b[k][l'] = \iota_b[k][l']$ para $k' > k$ e $l' > l$. Em outras palavras, os elementos maiores que k e todos os bits menos significativos do que $\pi_b[k][l]$ de $\pi[k]$ estão nas suas posições corretas.

Quando $i = N - 1$, somente é possível mudar $\pi[i]$ com $\pi[k']$, onde $k' < N - 1$, garantindo a satisfazibilidade da base de indução. A indução é garantida, se mostramos que a aplicação porta, no passo $i = k$, não afeta qualquer elemento $\pi[k']$, com $k' > k$.

No passo $i = k$ e $j = l$, se $\pi_b[k][l] = \iota_b[k][l]$ então nenhuma troca é necessária. Caso contrário, $\pi[k]$ deve alterar o l -ésimo bit com $\pi[m]$ tal que

$$\pi_b[m][p] = \begin{cases} \pi_b[k][p] & \text{se } p \neq l \\ !\pi_b[k][p] & \text{se } p = l, \end{cases}$$

onde $!x$ é 1, se $x = 0$ ou 0, caso contrário.

Se $\pi_b[k][l] = 1$ então a troca (de 1 para 0) é feita de um valor $\pi[m]$ menor que $\pi[k]$, por hipótese de indução, para um valor menor do que k . Se $\pi_b[k][l] = 0$, então a alteração é feita para um valor maior do que $\pi[k]$.

Agora devemos mostrar quando $\pi[m] < k$. De fato, tome o bit mais significativo l' de $\pi[k]$ tal que $\pi_b[k][l'] \neq \iota_b[k][l']$. Podemos separar em dois casos: (i) $l' = l$ (ii) $l' > l$. No caso (i), os bits mais significativos que l de $\pi[k]$ estão corretos pela hipótese de indução, os bits menos significativos que l de $\pi[k]$ estão corretos e $\pi[k'] = k'$ para $k' > k$, portanto se $m > k$ então $\pi[m] = k$ e $\pi[m] = k'$ não é possível. No caso (ii), se $m > k$ então $\pi_b[m][l'] = 1$, mas $\pi_b[m][l'] = \pi_b[k][l']$ onde $l' \neq l$, assim $\pi[k] > k$, que é contrário à hipótese de indução. \square

Teorema 4.8. *O circuito reversível retornado pelo Algoritmo 4.2 tem no máximo $(n - 1)2^n + 1$ portas Toffoli de controles mistos.*

Demonstração. Note que a aplicação de uma porta CM-Toffoli pelo Algoritmo 4.2, coloca pelo menos um bit em seu lugar. Portanto, o Algoritmo 4.2 termina após a aplicação de no máximo $n2^n$ portas CM-Toffoli. Para provar o limite superior, construímos uma função de pior caso para este algoritmo.

Após aplicações das primeiras $n2^{n-1}$ portas CM-Toffoli, considerando a aplicação da direita para esquerda, temos os últimos $n2^{n-1}$ bits da permutação de entrada coincidindo com os últimos bits da permutação de saída. Desta forma, temos a metade dos bits nos seus respectivos lugares. Após este passo, também temos o bit mais significativo completamente tratado, ou seja, a permutação resultante tem 2^{n-1} bits com valor zero nos primeiros 2^{n-1} elementos da permutação e os últimos $n2^{n-1}$ bits nas suas posições corretas. Portanto, após este passo o bit mais significativo está na sua posição.

A partir do primeiro passo, no máximo $(n-1)2^{n-1}$ bits ficam em posições erradas. Do mesmo modo que no primeiro passo, precisamos da aplicação de $(n-1)2^{n-2}$ portas CM-Toffoli para que o segundo bit mais significativo seja completamente tratado. Em geral, no passo i foram necessários $n2^{n-1} + (n-1)2^{n-2} \dots + n - (i-1)2^{n-i}$, para que o i -ésimo bit mais significativo seja completamente tratado. Assim, o número máximo de portas CM-Toffoli utilizadas pelo Algoritmo 4.2 será:

$$n2^{n-1} + (n-1)2^{n-2} + \dots + (n - (n-1))2^{n-n} = (n-1)2^n + 1.$$

Portanto, $(n - 1)2^n + 1$ é um limite superior para o tamanho do circuito obtido pelo Algoritmo 4.2. \square

Usando o Algoritmo 4.2, é possível encontrar uma permutação para qualquer valor de n que requeira no máximo $(n - 1)2^n + 1$ portas CM-Toffoli. Se o Algoritmo 4.2 é executado na ordem direita, então podemos encontrar a permutação [5 2 7 4 1 6 3 0] para $n = 3$ e a permutação [5 10 7 4 9 14 11 8 13 2 15 12 1 6 3 0] para $n = 4$. Se o Algoritmo 4.2 é executado na ordem esquerda, então podemos encontrar a permutação [7 4 1 6 3 0 5 2] para $n = 3$ e a permutação [15 12 9 14 3 0 13 2 7 4 1 6 11 8 5 10] para $n = 4$.

Tabela 4.5: Número de funções reversíveis 3×3 para o Algoritmo 4.2 usando portas CM-Toffoli e comparando com os resultados ótimos.

Tamanho	Número de permutações		
	Algoritmo 4.2 unidirecional	Algoritmo 4.2 bidirecional	resultados ótimos para a biblioteca L_M
17	1		
16	14		
15	92		
14	380	9	
13	1113	111	
12	2468	581	1
11	4311	1946	36
10	6083	4349	430
9	7044	6917	2408
8	6754	8255	7347
7	5379	7662	11756
6	3549	5546	10388
5	1922	3088	5472
4	839	1329	1903
3	286	424	476
2	72	90	90
1	12	12	12
0	1	1	1
média de portas	8.67	7.71	6.61

A Tabela 4.5 mostra os resultados da aplicação da versão do Algoritmo 4.2 sobre todas as $8! = 40320$ permutações quando $n = 3$. Nós mostramos o número de

permutações por cada contagem de porta, chamado de tamanho, e média das portas. A média foi calculada pela fórmula: $((\sum_{i=1}^{(n-1)2^n+1} i \cdot \text{NúmeroPermutações}) + 1)/8!$. Para cada cenário, mostramos o número médio de portas *CM*-Toffoli necessárias.

Usando a biblioteca L_M de portas *CM*-Toffoli encontramos a coluna de resultados ótimos da Tabela 4.5 através do Algoritmo 4.3 de busca em largura. O Algoritmo 4.3 encontra a menor distância entre o elemento identidade e todos os outros vértices. O Algoritmo 4.3 foi implementado em *PHP* e executando em um computador Intel Core2 Duo encontrou os resultados ótimos da Tabela 4.5 em aproximadamente $3h$, pois o algoritmo tem complexidade exponencial de $O(2^n! + (2^n! \cdot n2^{n-1})/2)$.

Algoritmo 4.3: Busca em largura

```

entrada:  $\iota$  vetor,  $F$  fila
saída :  $d$  vetor
1 início
2    $k = 0;$ 
3    $F = \text{InsereElementoFila}(\iota);$ 
4    $d[\iota] = k;$ 
5   enquanto  $F \neq \emptyset$  faça
6      $\text{vértice} = \text{RetiraElementoFila}(F);$ 
7      $k = d[\text{vértice}] + 1;$ 
8      $p = \text{CriaVizinhos}(\text{vértice});$ 
9     % utilizando a biblioteca  $L_M$  cria a lista de adjacência do
10    vértice
11    enquanto  $p \neq \emptyset$  faça
12       $u = \text{RetiraElementoFila}(p);$ 
13      se  $\text{BuscaVerticeMarcado}(u, d) = \emptyset$  então
14         $d[u] = k;$  % menor distância do vértice
15         $F = \text{InsereElementoFila}(u);$ 
16  fim

```

4.3.2 Método de controles mistos

O método de controles mistos para síntese de circuito reversível utiliza a biblioteca de porta L_M . Este método, apresentado no Algoritmo 4.4, usa aplicações consecutivas de portas *CM*-Toffoli para organizar os bits. O método proposto organiza o k -ésimo bit mais significativo a cada chamada recursiva do algoritmo. Para organizar cada bit mais significativo são necessários no máximo $n2^{n-1}$ trocas, ou seja, temos um limite superior de n^22^{n-1} trocas para organizar os n bits mais significativos. Estas trocas utilizam as portas *CM*-Toffoli. Veja o exemplo na Tabela 4.6. Os elementos que serão trocados são apresentados em negrito. Os elementos ordenados

são apresentados sublinhado. O circuito reversível correspondente é mostrado na Figura 4.13.

Tabela 4.6: Evolução da permutação π sendo transformada em ι pelo método de controles mistos.

porta aplicada	elementos da permutação							
	1	7	5	4	3	2	0	6
passo $i = 1$ $C^3NOT(a, b, c)$	001	111	101	100	011	010	000	110
	001	<u>011</u>	101	100	<u>111</u>	010	000	110
passo $i = 2$ $C^3NOT(a', b', c)$	001	011	101	100	111	010	000	110
	001	011	101	<u>000</u>	111	010	<u>100</u>	110
passo $i = 3$ $C^3NOT(b', c, a)$	001	011	101	000	111	010	100	110
	001	011	<u>100</u>	000	111	010	<u>101</u>	110
passo $i = 4$ $C^3NOT(a', c, b)$	001	011	100	000	111	010	101	110
	001	011	<u>110</u>	000	111	010	101	<u>100</u>
passo $i = 5$ $C^3NOT(a', b, c)$	001	011	110	000	111	010	101	100
	001	011	<u>010</u>	000	111	<u>110</u>	101	100
passo $i = 6$ $C^3NOT(b, c', a)$	001	011	010	000	111	110	101	100
	001	<u>010</u>	<u>011</u>	000	111	110	101	100
passo $i = 7$ $C^3NOT(a', c', b)$	001	010	011	000	111	110	101	100
	001	000	011	<u>010</u>	111	110	101	100
passo $i = 8$ $C^3NOT(b', c', a)$	001	000	011	010	111	110	101	100
	<u>000</u>	<u>001</u>	011	010	111	110	101	100
passo $i = 9$ $C^3NOT(b, c', a)$	000	001	011	010	111	110	101	100
	000	001	<u>010</u>	<u>011</u>	111	110	101	100
passo $i = 10$ $C^3NOT(a, c, b)$	000	001	010	011	111	110	101	100
	000	001	010	011	<u>101</u>	110	<u>111</u>	100
passo $i = 11$ $C^3NOT(a', c, b)$	000	001	010	011	101	110	111	100
	000	001	010	011	101	<u>100</u>	111	<u>110</u>
passo $i = 12$ $C^3NOT(b', c, a)$	000	001	010	011	101	100	111	110
	000	001	010	011	<u>100</u>	<u>101</u>	111	110
passo $i = 13$ $C^3NOT(b, c, a)$	000	001	010	011	100	101	111	110
	000	001	010	011	100	101	<u>110</u>	<u>111</u>
	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>

Por exemplo, a Tabela 4.6 mostra o método de controles mistos transformando a permutação $\pi = [1\ 7\ 5\ 4\ 3\ 2\ 0\ 6]$ na permutação identidade $\iota = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$.

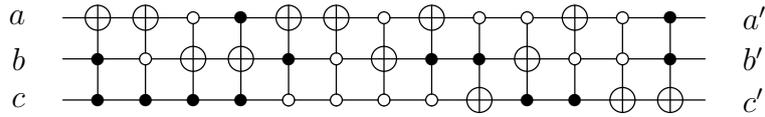


Figura 4.13: Circuito reversível que transforma a permutação ι na permutação $\pi = [1\ 7\ 5\ 4\ 3\ 2\ 0\ 6]$, de acordo com o exemplo da Tabela 4.6.

O Algoritmo 4.4 faz a síntese de circuito reversível usando as portas CM-Toffoli e considera as permutações ou as permutações inversas. Chamamos de *unidirecional* se o Algoritmo 4.4 é executado considerando ou a permutação ou a sua permutação inversa. Chamamos de *bidirecional* se o Algoritmo 4.4 é executado considerando a permutação e a sua permutação inversa, então escolhemos o menor circuito entre os dois.

Teorema 4.9. *O Algoritmo 4.4 retorna um circuito reversível usando portas CM-Toffoli.*

Demonstração. Provamos a correção do Algoritmo 4.4 por indução, mostrando que a aplicação de n portas CM-Toffoli coloca pelo menos um bit errado em seu lugar. Observe que inicialmente podemos ter até $n2^n$ bits em posições erradas. Dizemos que a permutação está organizada no b -ésimo bit, quando todos os elementos da metade esquerda do bloco tem os b -ésimo bits iguais a zero e da metade direita do bloco tem os b -ésimo bits iguais a um. Caso base: quando $n = 1$ e somente um bit está na posição errada, então o Algoritmo 4.4 simplesmente aplica uma porta NOT.

Hipótese de indução: se a permutação está organizada do $(b+1)$ -ésimo bit para o n -ésimo bit, então após a chamada recursiva a permutação está organizada também no b -ésimo bit. Assim, após n níveis de chamadas recursivas, a permutação será ordenada.

Por definição, a lista EE é composta de todos os elementos da permutação π tal que o b -ésimo bit esteja errado e posicionado na metade esquerda do bloco. Similarmente, a lista ED é composta de todos os elementos da permutação π tal que o b -ésimo bit esteja errado e posicionado na metade direita do bloco. Estas listas são preparadas no passo 1.

O passo 2 do algoritmo seleciona, para cada elemento da lista EE , um elemento da lista ED tal que tenha a menor distância de Hamming. Estes elementos são denotados por $\pi[\text{indice}_E]$ e $\pi[\text{indice}_D]$.

Algoritmo 4.4: Controles mistos

entrada: π , π_s e π_e , onde π é uma permutação, π_s é a posição inicial e π_e é a posição final

saída : *circuito* pilha

1 **início**

2 | **passo 0:** Procedimento Inicialização. Observe que este passo define os valores para b , N , n e π_m

3 | **passo 1:**

4 | **para** i de π_s até $\pi_m - 1$ **faça**

5 | - verifica se o b -ésimo bit de π_i está errado. Dizemos que um bit é errado quando difere do bit correspondente na permutação identidade ι_i . Sempre que encontramos o elemento π_i com o b -ésimo bit em uma posição errada, armazenamos o índice i correspondente em uma lista denotada por EE , que significa “lista errada esquerda”.

6 | - cópia EE para uma lista temporária EE_A .

7 | **para** i de π_m até π_e **faça**

8 | - verifica se o b -ésimo bit de π_i está errado. Analogamente, armazenar o índice i correspondente na lista denotada por ED , que significa “lista errada direita”.

9 | **passo 2:** Nesta etapa, temos três laços aninhados.

10 | **para** h de 1 até n **faça**

11 | **para** i de 0 até o tamanho da lista EE **faça**

12 | - inicializa as seguintes variáveis: $min \leftarrow \infty$; $indice_E \leftarrow \emptyset$ e $indice_D \leftarrow \emptyset$.

13 | **para** j de 0 até o tamanho da lista ED **faça**

14 | - calcular a distância de Hamming entre $\pi_{[EE_i]}$ e $\pi_{[ED_j]}$, denotamos por DH .

15 | **se** $DH < min$ **então**

16 | - atualiza as seguintes variáveis: $min \leftarrow DH$;

16 | - $indice_E \leftarrow EE_{A_i}$ e $indice_D \leftarrow ED_j$.

17 | **se** $min = h$ **então**

18 | **para** k de 0 até n **faça**

19 | - verifica se o k -ésimo bit em $\pi_{[indice_E]}$ e $\pi_{[indice_D]}$ são diferentes, se forem, aplica uma porta CM-Toffoli com alvo em k e controles correspondentes aos bits restantes em $\pi_{[indice_E]}$.

20 | - remova o elemento $indice_D$ de ED .

21 | - remova o elemento $indice_E$ de EE_A .

22 | - cópia EE_A para EE .

23 | **passo 3:**

24 | **se** $b > 0$ **então**

25 | - chame o Algoritmo 4.4 com entradas π , π_s e $\pi_m - 1$.

26 | - chame o Algoritmo 4.4 com entradas π , π_m e π_e .

27 **fim**

Procedimento Inicialização

1 início**2** | - calcule a posição corrente do bit de comparação como $\lceil \log(\pi_s - \pi_e) \rceil - 1 \equiv b$. Observe que b corresponde ao nível atual da recursão.**3** | - denotamos o tamanho de π como N .**4** | - calcule o número de bits de um elemento qualquer em π como $\log N \equiv n$.**5** | - calcule a posição do meio na permutação considerando o nível atual da recursão, denotamos como $\pi_m = \lfloor (\pi_e - \pi_s)/2 \rfloor$.**6 fim**

O laço mais interno do passo 3, faz a troca entre os elementos $\pi[indice_E]$ e $\pi[indice_D]$. Sabemos pela hipótese de indução, que para $k > b$ o k -ésimo bit já está organizado. Quando $i = indice_E$ e $j = indice_D$, se $\pi[i][k] = \pi[j][k]$, então nenhuma mudança é necessária. Caso contrário, o elemento $\pi[i]$ deve troca o k -ésimo bit com o elemento $\pi[j]$. Chamamos de h menor distância de Hamming entre $\pi[i]$ e $\pi[j]$, que organiza o k -ésimo.

Se $h = 1$, o algoritmo aplica uma porta CM-Toffoli com alvo em k e coloca o k -ésimo bit em sua posição correta.

Se $h > 1$, o algoritmo aplica h portas CM-Toffoli e o k -ésimo bit em sua posição correta. Devemos observar que: a) quaisquer k bit mais significativos do que o b -ésimo bit no elemento $\pi[i]$ estão organizados, tal que $\pi[i][k] = \pi[j][k]$; b) quaisquer k bit menos significativos do que o b -ésimo bit no elemento $\pi[i]$ podem estar desorganizados, tal que $\pi[i][k] \neq \pi[j][k]$. Portanto, o algoritmo aplica h portas CM-Toffoli de tal modo que os bits errados de (b) sejam organizados. \square

Teorema 4.10. *O circuito reversível retornado pelo Algoritmo 4.4 tem no máximo $T(n) = \lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$ portas CM-Toffoli.*

Demonstração. Seja $T(n)$ o número de portas CM-Toffoli necessárias para ordenar uma permutação P . Seja P uma permutação que tem a maior distância entre P e ι . A complexidade do Algoritmo 4.4 possui a seguinte relação de recorrência:

$$T(n) = S(n) + 2T(n - 1). \quad (4.1)$$

onde $S(n)$ é o número de portas CM-Toffoli necessárias para ordenar o bit mais significativo. Temos que $S(n) = \max(d(l, m))$, onde

$$d(l, m) = \begin{cases} 2^{l+m}(n - l) & \text{se } m = 0 \\ 2m(n - l - 1) + d(2^{l+m} - 2m) & \text{se } m \neq 0, \end{cases} \quad (4.2)$$

para $0 \leq l + m \leq n - 1$, $l \geq 0$, $m \geq 0$.

Então, temos que:

$$S(n) = \begin{cases} \sum_{j=0}^{\frac{n-1}{2}} 2^{2j} & \text{se } n \text{ é ímpar} \\ \sum_{j=0}^{\frac{n}{2}-1} 2^{2j+1} & \text{se } n \text{ é par,} \end{cases} \quad (4.3)$$

que é simplificada para

$$S(n) = \frac{2^{n+1}}{3} - \frac{1}{2} - \frac{(-1)^n}{6}. \quad (4.4)$$

Assim, a solução da Equação (4.1) é

$$T(n) = \frac{n2^{n+1}}{3} - \frac{2^{n+2}}{9} + \frac{(-1)^{n+1}}{18} + \frac{1}{2}. \quad (4.5)$$

Então temos que $T(n) = \lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$. Portanto, temos um limite superior para o tamanho do circuito obtido pelo Algoritmo 4.4. \square

A Tabela 4.7 mostra os resultados da aplicação da versão do Algoritmo 4.4 sobre todas as $8! = 40320$ permutações quando $n = 3$. Mostramos o número de funções para cada tamanho de circuito e a média de portas necessárias. Para cada cenário, mostramos o número médio de portas *CM*-Toffoli necessárias.

Tabela 4.7: Número de funções reversíveis 3×3 para o Algoritmo 4.4 usando portas *CM*-Toffoli e comparando com os resultados ótimos.

Tamanho	Número de permutações		
	Algoritmo 4.4 unidirecional	Algoritmo 4.4 bidirecional	resultados ótimos para a biblioteca L_M
13	36	2	
12	401	17	1
11	1776	328	36
10	4478	1958	430
9	7420	5560	2408
8	8783	9045	7347
7	7800	9784	11756
6	5300	7466	10388
5	2772	4026	5472
4	1119	1583	1903
3	344	448	476
2	78	90	90
1	12	12	12
0	1	1	1
média de portas	7.75	7.12	6.61

Usando o Algoritmo 4.4 podemos encontrar uma permutação para um valor qualquer de n que requeira no máximo $\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$ portas CM-Toffoli. Quando executamos o Algoritmo 4.4 na ordem unidirecional, encontramos a permutação $[6\ 0\ 5\ 4\ 3\ 2\ 7\ 1]$ para $n = 3$ e a permutação $[11\ 14\ 13\ 7\ 10\ 12\ 9\ 15\ 3\ 4\ 5\ 6\ 2\ 8\ 1\ 0]$ para $n = 4$.

4.4 Análise da síntese de circuito baseada no grafo de Cayley I_n

Nesta seção, apresentamos o grafo de Cayley I_n associado as portas G-Toffoli. Estas portas são usadas no método de síntese de circuito proposto por Miller, Maslov e Dueck [32, 33], como descrito na Seção 4.2.

Definição 4.11. C_I é um subconjunto de S_{2^n} , tal que todas as permutações $c \in C_I$ são L_I -construtíveis, com a aplicação de apenas uma porta G-Toffoli.

Lema 4.12. O subconjunto C_I é um conjunto gerador de S_{2^n} .

Demonstração. Imediato do Teorema 4.2. □

Seja (S_{2^n}, \cdot) o grupo simétrico com a operação de composição e C_I o conjunto gerador formado pelas portas G-Toffoli.

Teorema 4.13. O grafo $I_n(V, E)$ é o grafo de Cayley associado ao par $((S_{2^n}, \cdot), C_I)$.

Demonstração. Mostramos anteriormente que C_I é um conjunto gerador do grupo simétrico (S_{2^n}, \cdot) . Cada porta G-Toffoli é inversa dela mesma. A permutação identidade esta associada a não aplicar nenhuma porta, portanto não faz parte do conjunto gerador. Logo o grafo associado ao conjunto gerado C_I é o Grafo de Cayley $I_n(V, E)$. □

O grafo de Cayley I_n tem grau $n2^{n-1}$ e ordem $2^n!$. Para $n \leq 3$, os circuitos correspondentes têm n portas N , $n(n-1)$ portas C e $n(n-1)(n-2)/2$ portas T . Assim, para $i > 3$ existem $\binom{n}{i}$ portas G-Toffoli. Portanto,

$$\sum_{i=1}^n i \binom{n}{i} = 1 \binom{n}{1} + 2 \binom{n}{2} + 3 \binom{n}{3} + \dots + n \binom{n}{n} = n2^{n-1}.$$

Teorema 4.14. O limite superior para o diâmetro do grafo de Cayley I_n é $(n-1)2^n + 1$.

Demonstração. Segue diretamente da construção obtida no Algoritmo 4.1. □

Proposição 4.15. O grafo de Cayley I_n não é um grafo bipartido.

com alvo em b ; $(0, 1)(2, 3)$, que corresponde a porta $C^1CNOT(a)$ com alvo em a e $(2, 3)$, que corresponde a porta $C^2CNOT(a, b)$ com alvo em b . Portanto, temos um ciclo ímpar C_5 no grafo I_2 .

A Tabela 4.8 resume nossa análise de custo quântico para a síntese de circuito baseada no grafo de Cayley I_n , mostrando sua relação com o limite superior do diâmetro. A primeira coluna indica a quantidade de linhas de lixo. A segunda coluna indica o número de portas (np), que é o limite superior dado pelo diâmetro do grafo de Cayley I_n . A terceira coluna indica o custo quântico (cq) para a síntese de circuito baseada no grafo de Cayley I_n . O custo quântico é obtido através da multiplicação do diâmetro do grafo pelo custo quântico correspondente a cada porta mostrado na Tabela 4.1.

Tabela 4.8: Análise do custo quântico para a síntese de circuito baseada no grafo de Cayley I_n .

lixo	número de porta (np)	custo quântico (cq)
0	$(n - 1)2^n + 1$	$(2^n - 3)np$
1	$(n - 1)2^n + 1$	$(24n - 88)np$
n-3	$(n - 1)2^n + 1$	$(10n - 25)np$

4.5 Análise da síntese de circuito baseada no grafo de Cayley M_n

Nesta seção, apresentamos o grafo de Cayley M_n associado as portas CM-Toffoli. Estas portas são usadas no método Hipercubo de síntese de circuito. Método introduzido na Seção 4.3.

Definição 4.16. C_H é um subconjunto de S_{2^n} , tal que todas as permutações $c \in C_H$ são L_M -construtíveis, com a aplicação de apenas uma porta CM-Toffoli.

Lema 4.17. O subconjunto C_H é um conjunto gerador de S_{2^n} .

Demonstração. Imediato do Teorema 4.4. □

Seja (S_{2^n}, \cdot) o grupo simétrico com a operação de composição e C_H o conjunto gerador formado pelas portas CM-Toffoli.

Teorema 4.18. O grafo $M_n(V, E)$ é o grafo de Cayley associado ao par $((S_{2^n}, \cdot), C_H)$.

Demonstração. Mostramos anteriormente que C_H é um conjunto gerador do grupo simétrico (S_{2^n}, \cdot) . Cada porta CM-Toffoli é inversa dela mesma. A permutação

identidade, esta associada à não aplicar nenhuma porta, não faz parte do conjunto gerador. Logo o grafo associado ao conjunto gerado C_H é o Grafo de Cayley $M_n(V, E)$. \square

O grafo de Cayley M_n tem grau $n2^{n-1}$ e ordem $2^n!$. Nas portas CM-Toffoli podemos colocar o alvo em n linhas e os controles em $0 \leq k \leq 2^{n-1} - 1$, onde k é um valor decimal que representa as linhas de controles. Assim, temos $n2^{n-1}$ elementos no conjunto gerador C_H .

O grafo de Cayley M_n tem o seguinte limite inferior para a distância de um vértice para o vértice identidade ι .

Teorema 4.19. *O limite inferior para o distância de um vértice do grafo de Cayley M_n é $d(x, \iota) \geq d_H(x, \iota)/2$.*

Demonstração. Por definição, se aplicamos uma porta CM-Toffoli, temos as seguintes possibilidades: i) 2-movimentos, o que significa que dois bits simultaneamente errados ficaram em posições corretas; ii) 0-movimento, um bit vai para sua posição correta enquanto outro vai para uma posição errada; iii) -2-movimentos, quando dois bits simultaneamente certos ficaram em posições erradas. Temos o melhor caso quando o procedimento de ordenação utiliza apenas 2-movimentos. Portanto, temos $d(x, \iota) \geq d_H(x, \iota)/2$. \square

Teorema 4.20. *O limite superior para o diâmetro do grafo M_n é $\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$.*

Demonstração. Segue diretamente da construção obtida no Algoritmo 4.4. \square

A fim de obter um limite inferior para o diâmetro, vamos usar o fato que as portas CM-Toffoli alteraram as posições de dois bits.

Teorema 4.21. *O limite inferior para o diâmetro do grafo de Cayley M_n é $n2^{n-1}$.*

Demonstração. Observe que a permutação reversa tem distância de Hamming igual a $n2^n$ em relação a permutação identidade. Todas as portas CM-Toffoli aplicadas pelo Algoritmo 4.2 na permutação reversa é de 2-movimentos. Portanto, temos $n2^n/2 = n2^{n-1}$. \square

Tabela 4.9: Análise do custo quântico para a síntese de circuito baseada no grafo de Cayley M_n .

lixo	número de portas (np)	custo quântico (cq)
0	$\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$	$(2^n - 3 + 2m)\text{np}$
1	$\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$	$(24n - 86)\text{np}$
n-3	$\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$	$(10n - 23)\text{np}$

A Tabela 4.9 resume nossa análise de custo quântico para a síntese de circuito baseada no grafo de Cayley M_n , mostrando sua relação com o limite superior do diâmetro. A primeira coluna indica a quantidade de linhas de lixo. A segunda coluna indica o número de portas (np), que é o limite superior dado pelo diâmetro do grafo de Cayley M_n . A terceira coluna indica o custo quântico (cq) para a síntese de circuito baseada no grafo de Cayley M_n . O custo quântico é obtido através da multiplicação do diâmetro do grafo pelo custo quântico correspondente a cada porta mostrado na Tabela 4.1.

No próximo resultado usamos a notação de ciclo. Um ciclo (a_1, a_2, \dots, a_k) é uma permutação tal que $f(a_1) = a_2$, $f(a_2) = a_3$, $f(a_3) = a_4$, \dots , $f(a_k) = a_1$. Por exemplo, a permutação $[1\ 5\ 3\ 2\ 0\ 4\ 6\ 7]$ pode ser escrita como $(0, 1, 5, 4)(2, 3)(6)(7)$. O tamanho de um ciclo é o número de elementos do ciclo. Os ciclos c_1 e c_2 são disjuntos se eles não tem elementos em comum. Qualquer permutação pode ser escrita como o produto de ciclos disjuntos. A composição de dois ciclos disjuntos não depende da ordem em que os ciclos são aplicados. Chamamos de $tc(\pi)$ a quantidade de ciclos em um representação por ciclos de uma permutação.

Propriedade 4.22. *A aplicação de uma porta CM-Toffoli troca dois elementos na permutação e altera a quantidade de ciclos $tc(\pi)$ em dois casos:*

- (a) $tc(\pi') = tc(\pi) + 1$, quando um ciclo é dividido em dois ciclos;
- (b) $tc(\pi') = tc(\pi) - 1$, quando dois ciclos é unido em um ciclo.

Proposição 4.23. *O grafo de Cayley M_n é bipartido.*

Demonstração. Vamos provar que M_n é bipartido. Podemos dividir as permutações em dois conjuntos através da representação de ciclos da seguinte maneira:

$$X = \{v \in V(M_n) \mid tc(v) \text{ é par}\};$$

$$Y = \{v \in V(M_n) \mid tc(v) \text{ é ímpar}\}.$$

Vamos provar que $\{X, Y\}$ é uma bipartição de M_n . Para isso, vamos tomar dois vértices quaisquer u e v em X e provar que esses vértices não são adjacentes. Pela propriedade 4.22 temos que a aplicação de uma porta CM-Toffoli no vértice u obtém um vértice u' com paridade diferente para $tc(u)$ e $tc(u')$, ou seja, $tc(u') = tc(u) + 1$ ou $tc(u') = tc(u) - 1$. Como o vértice v tem a mesma paridade do vértice u , concluímos que não existe uma aresta entre os vértices u e v , ou seja, os vértices u e v não são adjacentes. Analogamente, conclui-se que quaisquer dois vértices de Y não são adjacentes. Portanto, $\{X, Y\}$ é uma bipartição de M_n , ou seja, M_n é bipartido. \square

Na Figura 4.15 apresentamos uma representação do grafo M_2 e uma coloração para os vértices com 2 cores.

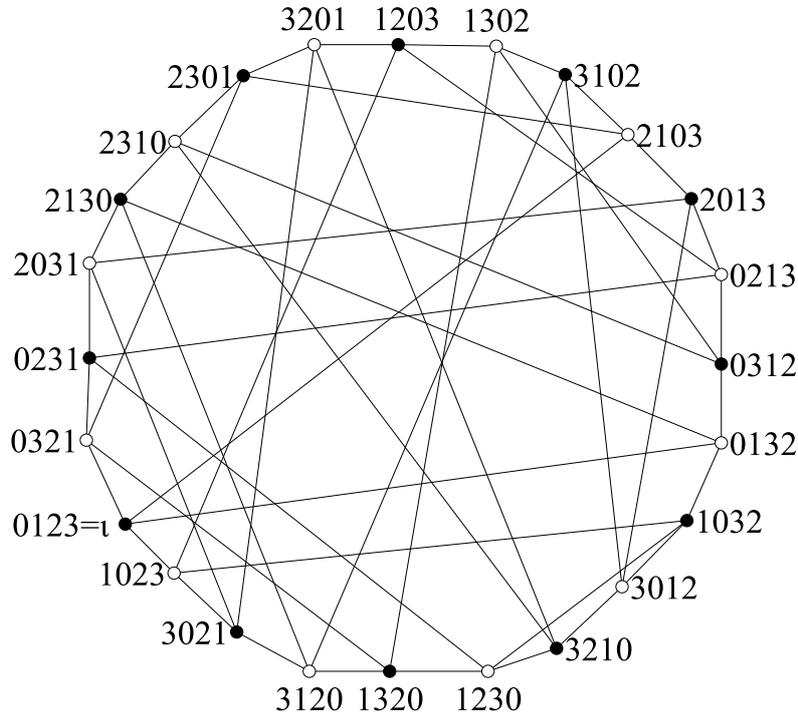


Figura 4.15: Grafo M_2 .

O grafo M_n tem o mesmo número de vértices e de elementos do conjunto gerador do grafo I_n , mas estes grafos de Cayley M_n e I_n não são isomorfos.

Teorema 4.24. *O grafo de Cayley M_n não é isomorfo ao grafo de Cayley I_n .*

Demonstração. Segue diretamente das Proposições 4.15 e 4.23. □

Vimos, neste capítulo, três métodos de síntese de circuitos reversíveis. O primeiro método foi baseado nos trabalhos de Miller, Maslov e Dueck [32, 33], este processo utiliza a biblioteca L_I composta das portas CNT. Os outros dois métodos proposto nesta tese utilizam a biblioteca L_M composta das portas Toffoli de controles mistos, com exatamente $n - 1$ controles. Utilizamos as bibliotecas L_I e L_M para definirmos dois grafos de Cayley, denotamos por grafos I_n e M_n . Depois, usamos as propriedades dos grafos de Cayley para analisar o custo quântico e a complexidade dos algoritmos destes três métodos de sínteses.

Capítulo 5

Conclusões

Concluimos esta tese com a esperança da utilização destes resultados nas diversas áreas de aplicabilidade deste estudo tais como: interconexões de redes, rearranjo de genoma e síntese de circuitos. Contribuímos para o estudo de importantes problemas relacionados a ciclo hamiltoniano, distância e diâmetro.

Primeiramente, mostramos que os grafos $H_{l,p}$ e $H'_{l,p}$ são grafos de Cayley, diâmetro na ordem de $O(pl)$ e grau logarítmicos em relação ao número de vértices. Portanto, considerando estas propriedades estabelecidas, propomos que os grafos de Cayley $H_{l,p}$ e $H'_{l,p}$ são bons esquemas para interconexão redes. Como o resultado de ciclos hamiltoniano nos grafos $H_{l,p}$ e $H'_{l,p}$ corroboramos a conjectura de Lovász.

Jwo [20] provou que o grafo *bubble-sort* $BS(n)$ é hamiltoniano. Como $BS(n)$ é um subgrafo gerador do grafo de rearranjo por transposições $TRG(n)$, a prova de Jwo implica que $TRG(n)$ também é hamiltoniano. Note que $BS(n)$ não é um subgrafo gerador do grafo $URG(n)$ (Veja Figura 3.2). Como o grafo $URG(n)$ é um outro subgrafo gerador do grafo $TRG(n)$, esta tese fornece uma prova alternativa para o problema de ciclo hamiltoniano do grafo $TRG(n)$, como também fornece evidências adicionais para a conjectura de Lovász. Na verdade, o grafo $URG(n)$ também é um subgrafo gerador do grafo de rearranjo por transposições pré-fixadas $PRG(n)$ [12]. Além disso, a prova contribui para o estudo do problema de ordenação por transposições, fornecendo uma lista de todas as permutações de um determinado tamanho, em uma ordem que tem uma transposição pré-fixada unitária entre duas permutações consecutivas, e todas as permutações onde os mesmos últimos elementos são consecutivos.

A Tabela 5.1 mostra os resultados e os problemas em abertos que estão relacionados com as variações dos grafos de rearranjo por transposições. O grau de cada vértice é regular e denotamos por Δ . Observe que, para os casos esparsos $URG(n)$ e $BS(n)$, o diâmetro é conhecido. Surpreendentemente, o grafo $PRG(n)$ é mais esparsa do que o grafo $TRG(n)$, e ainda o possível intervalo dado pelos limites para o diâmetro é muito maior. Acreditamos que o presente trabalho no grafo $URG(n)$

contribui para o estudo do grafo $PRG(n)$.

Tabela 5.1: Resultados conhecidos dos grafos $TRG(n)$, $PRG(n)$, $URG(n)$ e $BS(n)$.

Grafo	Δ	Problema da distância	Diâmetro	Problema do ciclo hamiltoniano
$TRG(n)$	$\Theta(n^3)$	proposto por [3] e resolvido \mathcal{NP} -difícil por [7]	limite inferior e superior em [3] $\frac{n}{2} \leq D \leq \frac{3n}{4}$	resolvido com um corolário por [20] e resolvido com um corolário desta tese
$PRG(n)$	$\Theta(n^2)$	proposto por [12] e complexidade não conhecida	limite inferior por [12] e limite superior por [9] $\frac{n}{2} \leq D \leq n - \log_{\frac{9}{2}} n$	resolvido como um corolário desta tese
$URG(n)$	$\Theta(n)$	complexidade não conhecida	resolvido nesta tese $D = n - 1$	resolvido nesta tese
$BS(n)$	$\Theta(n)$	resolvido por [10]	resolvido por [2] $D = n(n - 1)/2$	resolvido por [20]

Uma vez que a reversibilidade é considerada um aspecto essencial para a modelagem de circuitos para computadores quânticos, são necessários métodos mais eficientes nas construções e análises dos circuitos reversíveis. A teoria dos grupos fornece uma estrutura unificada para a análise dos métodos de sínteses de circuitos reversíveis. Nesta tese, foram estudados dois grafos de Cayley, I_n e M_n , estes grafos são aplicados nas análises de sínteses de circuitos reversíveis.

Miller, Maslov e Dueck [32, 33] propuseram um algoritmo para síntese de circuito reversível usando as portas Toffoli generalizadas, estas portas modelam o grafo de Cayley I_n . A estrutura do grafo de Cayley permitiu comprovar que o diâmetro do grafo I_n é no máximo $(n - 1)2^n + 1$ e o número de vértices é de $2^n!$. Estes limites são consistentes com a contagem de portas e complexidade do custo quântico do processo de síntese de circuito usando portas G-Toffoli.

Apresentamos um algoritmo para a síntese de circuito reversível usando as portas Toffoli de controles mistos. A síntese de circuito proposta é modelada no grafo de Cayley M_n . O diâmetro do grafo de Cayley M_n é de no máximo $\lfloor \frac{2}{3}2^n(n - \frac{2}{3}) \rfloor + 1$ e pelo menos $n2^{n-1}$. Uma vez que o número de vértices do grafo M_n é de $2^n!$, temos que o número de vértices é fatorial do diâmetro. Provamos que o grafo M_n não é isomorfo ao grafo I_n , portanto os algoritmos correspondentes de síntese são diferentes. Estes limites são consistentes com a contagem de portas e complexidade do custo quântico no processo de síntese de circuito usando portas

CM-Toffoli. Portanto, quando modelamos as sínteses de circuitos através dos grafos de Cayley obtemos um formalismo teórico que auxilia as análises dos processos de sínteses circuitos reversíveis.

5.1 Trabalhos futuros

No decorrer dos estudos desta tese encontramos em vários momentos benefícios na utilização das propriedades dos grafos de Cayley na implementação dos algoritmos. Como não temos todas as respostas sobre o tema, apresentamos a seguir uma lista de algumas perguntas de nosso interesse que ainda não foram respondidas no decorrer deste trabalho:

- encontrar o valor exato do diâmetro do grafo $H_{l,p}$;
- provar a conjectura de Lovász, ou encontrar um grafo vértice-transitivo que não possui caminho hamiltoniano;
- desenvolver um algoritmo para encontrar ciclos hamiltonianos nos grafos de Cayley do grupo simétrico S_n utilizando espaço de memória de apenas $O(n)$;
- propor um método ótimo de síntese de circuitos para a biblioteca L_M , ou mostrar que o problema é \mathcal{NP} -difícil;

Referências Bibliográficas

- [1] Aigner, M., West, D. B., 1987, “Sorting by insertion of leading elements”, *J. Combin. Theory Ser. A*, v. 45, n. 2, pp. 306–309.
- [2] Akers, S. B., Krishnamurthy, B., 1989, “A group-theoretic model for symmetric interconnection networks”, *IEEE Trans. Comput.*, v. 38, pp. 555–565.
- [3] Bafna, V., Pevzner, P. A., 1998, “Sorting by transpositions”, *SIAM J. Discrete Math.*, v. 11, pp. 224–240.
- [4] Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A., Weinfurter, H., 1995, “Elementary gates for quantum computation”, *Phys. Rev. A*, v. 52, n. 5 (Nov.), pp. 3457–3467.
- [5] Biggs, N., 1993, *Algebraic Graph Theory*. 2nd ed. Cambridge, UK, Cambridge University Press.
- [6] Bóna, M., 2012, *Combinatorics of Permutations*. Boca Raton, Florida, The CRC Press.
- [7] Bulteau, L., Fertin, G., Rusu, I., 2012, “Sorting by Transpositions Is Difficult”, *SIAM J. Discrete Math.*, v. 26, n. 3, pp. 1148–1180.
- [8] Caprara, A., 1999, “Sorting Permutations by Reversals and Eulerian Cycle Decompositions”, *SIAM J. Discrete Math.*, v. 12, n. 1, pp. 91–110.
- [9] Chitturi, B., Sudborough, I. H., 2012, “Bounding prefix transposition distance for strings and permutations”, *Theoret. Comput. Sci.*, v. 421 (Mar.), pp. 15–24.
- [10] Darlington, J., 1978, “A synthesis of several sorting algorithms”, *Acta Inform.*, v. 11, pp. 21–30.
- [11] Devos, A., Raa, B., Storme, L., 2002, “Generating the group of reversible logic gates”, *J. Phys. A*, v. 35 (Aug.), pp. 7063–7078.

- [12] Dias, Z., Meidanis, J., 2002, “Sorting by prefix transpositions”. In: *String Processing and Information Retrieval (SPIRE), Lecture Notes in Computer Science*, v. 2476, pp. 463–468. Springer-Verlag.
- [13] Eriksson, H., Eriksson, K., Karlander, J., Svensson, L., Wästlund, J., 2001, “Sorting a bridge hand”, *Discrete Math.*, v. 241, n. 1, pp. 289–300.
- [14] Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S., 2009, *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. London, England, MIT Press.
- [15] Fortuna, V. J., Meidanis, J., 2004, *Sorting the Reverse Permutation by Prefix Transpositions*. Relatório Técnico IC-04-04, Instituto de Computação, Universidade Estadual de Campinas, Campinas, Brazil, Apr.
- [16] Gray, F., 1953, “Pulse code communication”. In: *USA Patent*, n. 2,632,058.
- [17] Hausen, R. A., Faria, L., Figueiredo, C. M. H., Kowada, L. A. B., 2010, “Unitary Toric Classes, the Reality and Desire Diagram, and Sorting by Transpositions”, *SIAM J. Discrete Math.*, v. 24, n. 3 (Jul.), pp. 792–807.
- [18] Holyer, I., 1981, “The NP –Completeness of Some Edge-Partition Problems”, *SIAM J. Comput.*, v. 10, n. 4, pp. 713–717.
- [19] Jiang, M., Ruskey, F., 1994, “Determining the hamilton-connectedness of certain vertex-transitive graphs”, *Discrete Math.*, v. 133, n. 1–3, pp. 159 – 169.
- [20] Jwo, J.-S., 1991, *Analysis of interconnection networks based on Cayley graphs related to permutation groups*. Tese de Doutorado, The University of Oklahoma.
- [21] Karp, R. M., 1972, “Reducibility Among Combinatorial Problems”. In: *Complexity of Computer Computations*, pp. 85–103, New York. Plenum Press.
- [22] Knuth, D. E., 2005, *The art of computer programming*, v. 4. United States, Pearson Education, Inc.
- [23] Konstantinova, E., 2008, “Some problems on Cayley graphs”, *Linear Algebra Appl.*, v. 429, n. 11-12, pp. 2754–2769.
- [24] Kowada, L. A. B., 2006, *Construção de Algoritmos Reversíveis e Quânticos*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, Programa de Engenharia de Sistemas e Computação, Rio de Janeiro.

- [25] Labarre, A., 2008, “Edit distances and factorisations of even permutations”. In: *16th European Symposium on Algorithms (ESA), Lecture Notes in Comput. Sci.*, v. 5193, pp. 635–646. Springer-Verlag.
- [26] Lakshmivarahan, S., Jwo, J.-S., Dhall, S. K., 1993, “Symmetry in Interconnection Networks Based on Cayley Graphs of Permutation Groups: A Survey”, *Parallel Comput.*, v. 19, n. 4, pp. 361–407.
- [27] Landauer, R., 1961, “Irreversibility and heat generation in the computing process”, *IBM J. Res. Develop.*, v. 5, pp. 183–191.
- [28] Lovász, L., 1970, “Problem 11”. In: *Combinatorial Structures and their Applications*, p. 497. Gordon and Breach.
- [29] Marušič, D., 1983, “Hamiltonian circuits in Cayley graphs”, *Discrete Math.*, v. 46, n. 1, pp. 49–54.
- [30] Maslov, D., Saeedi, M., 2011, “Reversible Circuit Optimization Via Leaving the Boolean Domain”, *IEEE T. Comput. Aid. D.*, v. 30, n. 6 (Jun.), pp. 806–816.
- [31] Maslov, D., Young, C., Miller, D. M., Dueck, G. W., 2005, “Quantum circuit simplification using templates”. In: *Proceedings of Design, Automation and Test in Europe*, v. 2, pp. 1208–1213.
- [32] Maslov, D., Dueck, G. W., Miller, D. M., 2005, “Toffoli network synthesis with templates”, *IEEE T. Comput. Aid. D.*, v. 24, n. 6 (Jun.), pp. 807–817.
- [33] Miller, D. M., Maslov, D., Dueck, G. W., 2003, “A transformation based algorithm for reversible logic synthesis”. In: *Proceedings of the 40th annual Design Automation Conference*, pp. 318–323, New York, NY, USA. ACM.
- [34] Nielsen, M. A., Chuang, I. L., 2000, *Quantum Computation and Quantum Information*. Cambridge, UK, Cambridge University Press.
- [35] Reis, C. S., 2010, *Ciclo Hamiltoniano em Grafos de Rearranjo de Genomas por Transposições Pré-Fixadas*. Dissertação de mestrado, UFRJ, Brasil.
- [36] Reis, C. S., Ribeiro, A. C., Kowada, L. A. B., Bueno, L. R., Figueiredo, C. M. H., 2013, “Hamiltonian Cycles in Unitary Prefix Transposition Rearrangement Graphs”, Submetido para publicação no *Discrete Appl. Math.*, Mar.

- [37] Ribeiro, A. C., de Figueiredo, C. M. H., Marquezino, F. L., Kowada, L. A. B., 2012, “Cayley graphs and analysis of quantum cost for reversible circuit synthesis”. In: *IV Workshop-School on Quantum Computation and Information WECIQ*.
- [38] Ribeiro, A. C., Kowada, L. A. B., , Figueiredo, C. M. H., 2013, “Two Families of Cayley Graph Interconnection Networks”, Submetido para publicação na *Mat. Contemp.*, Abr.
- [39] Ribeiro, A. C., Marquezino, F. L., Kowada, L. A. B., Junior, C. S. S., Figueiredo, C. M. H., 2013, “A new reversible circuit synthesis algorithm and a theoretical formalism using Cayley graphs”, Artigo completo a ser submetido para publicação no *ACM J. Emerg. Technol. Comput. Syst.*, Set.
- [40] Ribeiro, A. C., Figueiredo, C. M. H., Kowada, L. A. B., 2010, “An evidence for Lovász conjecture about Hamiltonian paths and cycles”, *Mat. Contemp.*, v. 39, pp. 121–128.
- [41] Saeedi, M., Markov, I. L., 2013, “Synthesis and optimization of reversible circuits - a survey”, *ACM Comput. Surv.*, v. 45, n. 2 (Mar.), pp. 21:1–21:34.
- [42] Shende, V. V., Prasad, A. K., Markov, I. L., Hayes, J. P., 2003, “Synthesis of reversible logic circuits”, *IEEE T. Comput. Aid. D.*, v. 22, n. 6, pp. 710–722.
- [43] Toffoli, T., 1980, “Reversible Computing”. In: *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pp. 632–644, London, UK, UK. Springer-Verlag.
- [44] Vadapalli, P., Srimani, P. K., 1996, “A New Family of Cayley Graph Interconnection Networks of Constant Degree Four”, *IEEE Trans. Parallel Distrib. Systems*, v. 7, n. 1 (Jan.), pp. 26–32.
- [45] Wille, R., Saeedi, M., Drechsler, R., 2009, “Synthesis of Reversible Functions Beyond Gate Count and Quantum Cost”. In: *International Workshop on Logic and Synthesis*.